

Factorisation of Greedoid Polynomials of Rooted Digraphs

Kai Siong Yow^{*1}, Kerri Morgan^{†2}, and Graham Farr^{‡1}

¹Faculty of Information Technology, Monash University, Clayton,
Victoria 3800, Australia

²Deakin University, Geelong, Australia, School of Information
Technology, Faculty of Science Engineering & Built Environment

September 9, 2018

Abstract

Gordon and McMahon defined a two-variable greedoid polynomial $f(G; t, z)$ for any greedoid G . They studied greedoid polynomials for greedoids associated with rooted graphs and rooted digraphs. They proved that greedoid polynomials of rooted digraphs have the multiplicative direct sum property. In addition, these polynomials are divisible by $1 + z$ under certain conditions. We compute the greedoid polynomials for all rooted digraphs up to order six. A greedoid polynomial $f(D)$ of a rooted digraph D of order n *GM-factorises* if $f(D) = f(G) \cdot f(H)$ such that G and H are rooted digraphs of order at most n and $f(G), f(H) \neq 1$. We study the GM-factorability of greedoid polynomials of rooted digraphs, particularly those that are not divisible by $1 + z$. We give some examples and an infinite family of rooted digraphs that are not direct sums but their greedoid polynomials GM-factorise.

Keywords: factorisation, greedoid polynomial, greedoid, directed branching greedoid, rooted digraph, arborescence

1 Introduction

Greedoids were introduced by Korte and Lovász as collections of sets that generalise matroids [11]. Korte and Lovász observed that the optimality of some “greedy” algorithms including breadth-first search could be traced back to an underlying combinatorial structure that satisfies the greedoid, but not the matroid, framework. Björner

^{*}Email: KaiSiong.Yow@monash.edu

[†]Email: Kerri.Morgan@deakin.edu.au

[‡]Email: Graham.Farr@monash.edu

and Ziegler [1] used two algorithmic constructions of a minimum spanning tree of a connected graph, i.e., Kruskal’s and Prim’s algorithms, to distinguish between greedoids and matroids. For each step in both algorithms, an edge with the minimum weight is added into the minimum spanning tree. The edge sets of the trees/forests that are obtained in each step form the feasible sets of a greedoid. Feasible sets obtained via Kruskal’s algorithm remain feasible when removing any edge from the sets. However, this is not always true for feasible sets that are obtained via Prim’s algorithm. Therefore, the greedoid that is obtained by using Kruskal’s algorithm (but not Prim’s algorithm) is in fact a matroid.

There are two equivalent ways to define greedoids, using set systems or hereditary languages [13, 14]. We define greedoids based on set systems. A *greedoid* over a finite ground set E is a pair (E, F) where $F \subseteq 2^E$ is a collection of subsets of E (called the *feasible sets*) satisfying:

- (G1) For every non-empty $X \in F$, there is an element $x \in X$ such that $X - \{x\} \in F$.
- (G2) For $X, Y \in F$ with $|X| < |Y|$, there is an element $y \in Y - X$ such that $X \cup \{y\} \in F$.

The *rank* $r(A)$ of a subset $A \subseteq E$ in a greedoid (E, F) is defined as $r(A) = \max\{|X| : X \subseteq A, X \in F\}$. Any greedoid is uniquely determined by its rank function.

Theorem 1.1. [12] *A function $r : 2^E \mapsto \mathbb{N} \cup \{0\}$ is the rank function of a greedoid (E, F) if and only if for all $X, Y \subseteq E$ and for all $x, y \in E$ the following conditions hold:*

- (R1) $r(X) \leq |X|$.
- (R2) If $X \subseteq Y$, then $r(X) \leq r(Y)$.
- (R3) If $r(X) = r(X \cup \{x\}) = r(X \cup \{y\})$, then $r(X) = r(X \cup \{x\} \cup \{y\})$.

Important classes of greedoids are those associated with rooted graphs and rooted digraphs. These are called *branching greedoids* and *directed branching greedoids*, respectively. We focus on directed branching greedoids. Hence, all our digraphs are rooted.

An *arborescence* [20] is a directed tree rooted at a vertex v such that every edge that is incident with v is an outgoing edge, and exactly one edge is directed into each of the other vertices. For every non-root vertex in an arborescence, there exists a unique directed path in the arborescence that leads from the root vertex to the non-root vertex. Occasionally, to highlight this property, people describe the root vertex as *Rome*¹ [20]. Some authors define arborescences by reversing the direction of each edge in our definition, giving a set of arborescences that is different to ours. In this scenario, each unique directed path in the arborescence directs into rather than away from the root vertex. In both definitions, the number of arborescences rooted at each

¹From the proverb: All roads lead to Rome.

vertex is identical. To change from one definition to the other, simply reverse the direction for all the edges.

Let D be a rooted digraph. A subdigraph F of D is *feasible* if F is an arborescence. We call the edge set of F a *feasible set*. If the edge set of F is maximal, then it is a *basis*. A *spanning arborescence* of D is a subdigraph of D that is an arborescence which includes every vertex of D . The *rank* of a subset $X \subseteq E(D)$ is defined as $r(X) = \max\{|A| : A \subseteq X, A \text{ is feasible}\}$.

A *directed branching greedoid* over a finite set E of directed edges of a rooted digraph is a pair (E, F) where F is the set of feasible subsets of E . This was defined and shown to be a greedoid by Korte and Lovász [12].

Let G be a greedoid. Gordon and McMahon [8] defined a two-variable greedoid polynomial of G

$$f(G; t, z) = \sum_{A \subseteq E(G)} t^{r(G)-r(A)} z^{|A|-r(A)}$$

which generalises the one-variable greedoid polynomial $\lambda(G; t)$ given by Björner and Ziegler in [1]. We call the two-variable greedoid polynomial $f(G; t, z)$ the *greedoid polynomial*. The greedoid polynomial is motivated by the Tutte polynomial of a matroid [19], and is an analogue of the Whitney rank generating function [21]. This polynomial is one of the digraph polynomials that is analogous of the Tutte polynomial. A survey of such polynomials for directed graphs can be found in [4].

Gordon and McMahon studied greedoid polynomials for branching greedoids and directed branching greedoids. They showed that $f(D; t, z)$ can be used to determine if a rooted digraph D is a rooted arborescence [8]. However, this result does not hold when D is an unrooted tree [6].

Suppose D , D_1 and D_2 are rooted digraphs, and $E(D_1), E(D_2) \subseteq E(D)$. The digraph D is the *direct sum* of D_1 and D_2 , if $E(D_1) \cup E(D_2) = E(D)$, $E(D_1) \cap E(D_2) = \emptyset$ and the feasible sets of D are precisely the unions of feasible sets of D_1 and D_2 . Gordon and McMahon proved that the greedoid polynomials of rooted digraphs have the multiplicative direct sum property, that is, if D is the direct sum of D_1 and D_2 , then $f(D; t, z) = f(D_1; t, z) \cdot f(D_2; t, z)$. This raises the question of whether this is the only circumstance in which this polynomial can be factorised. The Tutte polynomial of a graph G factorises if and only if G is a direct sum [16], but the situation for the chromatic polynomial is more complex [17].

Gordon and McMahon showed that the greedoid polynomial of a rooted digraph that is not necessarily a direct sum has $1+z$ among its factors under certain conditions (see Theorem 1.3 and 1.4). We address more general types of factorisation in this article.

Gordon and McMahon gave a recurrence formula to compute $f(D; t, z)$ where D is a rooted digraph. The following proposition gives the formula, which involves the usual deletion-contraction operations.

Proposition 1.2. [8] *Let D be a digraph rooted at a vertex v , and e be an outgoing*

edge of v . Then

$$f(D; t, z) = f(D/e; t, z) + t^{r(D)-r(D\setminus e)} f(D \setminus e; t, z).$$

A *greedoid loop* [15] in a rooted graph, or a rooted digraph, is an edge that is in no feasible set. It is either an ordinary (directed) loop, or an edge that belongs to no (directed) path from the root vertex.

Theorem 1.3. [15] *Let D be a rooted digraph that has no greedoid loops. Then D has a directed cycle if and only if $1 + z$ divides $f(D)$.*

Let G be a greedoid. A subset $S \subseteq E(G)$ is *spanning* if S contains a basis. Gordon and McMahon gave a graph-theoretic interpretation for the highest power of $1 + z$ which divides $f(G)$ in the following theorem.

Theorem 1.4. [9] *Let G be the directed branching greedoid associated with a rooted digraph D with no greedoid loops or isolated vertices. If $f(G; t, z) = (1 + z)^k h(t, z)$, where $1 + z$ does not divide $h(t, z)$, then k is the minimum number of edges that need to be removed from D to leave a spanning acyclic directed graph.*

Tedford [18] defined a three-variable greedoid polynomial $f(G; t, p, q)$ for any finite rooted graph G , which generalises the two-variable greedoid polynomial. He showed that $f(G; t, p, q)$ obeys a recursive formula. He also proved that $f(G; t, p, q)$ determines the number of greedoid loops in any rooted graph G . His main result shows that $f(G; t, p, q)$ distinguishes connected rooted graphs G that are loopless and have at most one cycle. He extended $f(G; t, p, q)$ from rooted graphs to general greedoids, and proved that the polynomial determines the number of loops for a larger class of greedoids.

In this article, we compute the greedoid polynomials for all rooted digraphs (up to isomorphism unless otherwise stated) up to order six. All the labelled rooted digraphs (without loops and multiple edges, but cycles of size two are allowed) up to order six were provided by Brendan McKay² on 28 March 2018 (personal communication from McKay to Farr). We then study the factorability of these polynomials, particularly those that are not divisible by $1 + z$.

Two rooted digraphs are *GM-equivalent* if they both have the same greedoid polynomial. If a rooted digraph is a direct sum, then it is *separable*. Otherwise, it is *non-separable*.

A greedoid polynomial $f(D)$ of a rooted digraph D of order n *GM-factorises* if $f(D) = f(G) \cdot f(H)$ such that G and H are rooted digraphs of order at most n and $f(G), f(H) \neq 1$. Note that $f(G)$ and $f(H)$ are not necessarily distinct. The polynomials $f(G)$ and $f(H)$ are *GM-factors* of $f(D)$. We also say a rooted digraph D *GM-factorises* if its greedoid polynomial GM-factorises. Every rooted digraph that is a direct sum has a GM-factorisation.

An irreducible GM-factor is *basic* if the GM-factor is either $1+t$ or $1+z$. Otherwise, the irreducible GM-factor is *nonbasic*. We are most interested in nonbasic GM-factors.

²More combinatorial data can be found at <https://users.cecs.anu.edu.au/~bdm/data/>.

A GM-factor is *primary* if it is irreducible, nonbasic and is not a GM-factor of any greedoid polynomial of rooted digraphs of smaller order. Such a factor appears as a GM-factor only in rooted digraphs with at least as many vertices as the current one. For $k \geq 1$, a non-separable digraph is *k-nonbasic* if its greedoid polynomial has k nonbasic GM-factors. A non-separable digraph is *totally k-nonbasic* if it is k -nonbasic and contains no basic GM-factors. Likewise, a non-separable digraph is *k-primary* if its greedoid polynomial has k primary GM-factors. A non-separable digraph is *totally k-primary* if it is k -primary and contains no basic GM-factors. It follows that if a non-separable digraph is (totally) k -primary, then the digraph is (totally) k -nonbasic.

Our results show that there exist non-separable digraphs that GM-factorise and their polynomials have neither $1 + t$ nor $1 + z$ as factors. In some cases (but not all), these non-separable digraphs of order n are GM-equivalent to a separable digraph of order at most n . We give the numbers of polynomials of this type of non-separable digraph. For rooted digraphs up to order six and $k \geq 2$, we found that there exist no $(k + 1)$ -nonbasic digraphs and no k -primary digraphs. We also provide the numbers of 2-nonbasic digraphs, totally 2-nonbasic digraphs, 1-primary digraphs and totally 1-primary digraphs. We then give the first examples of totally 2-nonbasic and totally 1-primary digraphs. Lastly, we give an infinite family of non-separable digraphs where their greedoid polynomials factorise into at least two non-basic GM-factors.

2 Results

The greedoid polynomials of all rooted digraphs up to order six were computed by using Algorithm 1 (see [Appendix](#)). This algorithm is based on the deletion-contraction recurrence in Proposition 1.2 that was introduced by Gordon and McMahon [8]. We then simplified and factorised all these greedoid polynomials using Wolfram Mathematica.

Throughout, rooted digraphs are up to isomorphism unless stated otherwise.

2.1 Separability and Non-separability

For each order, we determined the numbers of rooted digraphs, separable digraphs, non-separable digraphs, and non-separable digraphs of order n that are GM-equivalent to some separable digraph of order at most n (see Table 2, and the list of abbreviations in Table 1).

Note that the sequences of numbers of labelled rooted digraphs (T) and rooted digraphs (T-ISO) are not listed in The On-Line Encyclopedia of Integer Sequences (OEIS).

We observe that the ratio of T-ISO to T shows an increasing trend. The ratio of NS to T-ISO is also increasing (for $n \geq 3$), as expected.

Abbreviation	Description
T	Number of labelled rooted digraphs
T-ISO	Number of rooted digraphs
S	Number of separable digraphs
NS	Number of non-separable digraphs
NSE	Number of non-separable digraphs of order n that are GM-equivalent to some separable digraph of order at most n

Table 1: Abbreviations for Table 2

n	T	T-ISO	S	NS	NSE
1	1	1	0	1	0
2	6	4	0	4	0
3	48	36	6	30	7
4	872	752	88	664	200
5	48040	45960	2404	43556	10641
6	9245664	9133760	150066	8983694	1453437

Table 2: Numbers of various types of rooted digraphs (up to order six)

For each order, we also provide the number PU of unique greedoid polynomials and the ratio of PU to T-ISO, in Table 3.

n	T-ISO	PU	PU/T-ISO
1	1	1	1.0000
2	4	4	1.0000
3	36	22	0.6111
4	752	201	0.2673
5	45960	6136	0.1335
6	9133760	849430	0.0930

Table 3: Numbers PU of unique greedoid polynomials of rooted digraphs (up to order six) and the ratio of PU to T-ISO

Bollobás, Pebody and Riordan conjectured that almost all graphs are determined by their chromatic or Tutte polynomials [2]. However, this conjecture does not hold for matroids. The ratio of the number of unique Tutte polynomials of matroids to the number of non-isomorphic matroids approaches 0 as the cardinality of matroids increases, which can be shown using the bounds given in Exercise 6.9 in [3]. We believe that greedoid polynomials of rooted digraphs behave in a similar manner as matroids. According to our findings, the ratio of PU to T-ISO shows a decreasing trend. We expect that as n increases, this ratio continues to decrease. The question

is, does this ratio eventually approach 0 or is it bounded away from 0? Further computation should give more insight on this question.

2.2 Factorability

For $n \in \{1, \dots, 5\}$, we identified the numbers of greedoid polynomials that GM-factorise for rooted digraphs of order n . Details are given in Table 5 (see Table 4 for the list of abbreviations and Figure 1 for the corresponding Venn diagram).

Abbreviation	Description
	Number of unique greedoid polynomials of rooted digraphs that ...
PNF	... cannot be GM-factorised
PF	... can be GM-factorised
PFS	... can be GM-factorised and the digraph is separable
PFNS	... can be GM-factorised and the digraph is non-separable
PF	$PFS \cup PFNS$
COMM	$PFS \cap PFNS$
PFSU	$PFS - COMM$
PFNSU	$PFNS - COMM$

Table 4: Abbreviations for Figure 1 and Table 5

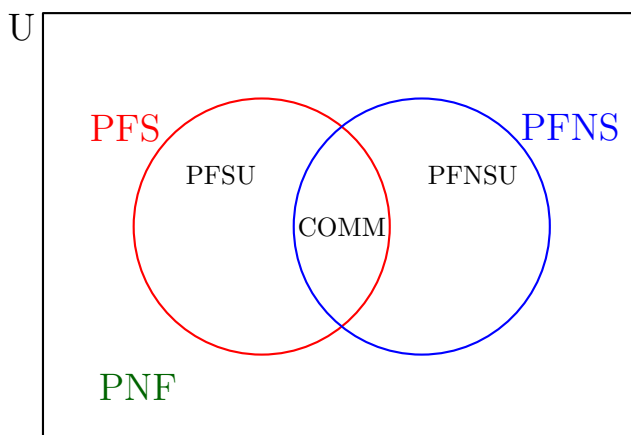


Figure 1: Venn diagram that represents the factorability of greedoid polynomials of rooted digraphs where $U = PF \cup PNF$ and $PF = PFS \cup PFNS$

n	PNF	PF	PFS	PFNS	COMM	PFSU	PFNSU
1	1	0	0	0	0	0	0
2	3	1	0	1	0	0	1
3	6	16	6	13	3	3	10
4	37	164	41	145	22	19	123
5	1044	5092	444	4867	219	225	4648

Table 5: Factorability of greedoid polynomials of rooted digraphs (up to order five)

We found that the ratio of PF to PU shows an upward trend, and the ratio stands at 0.8299 when $n = 5$. It seems that most likely as n increases, the ratio will either approach 1 in which case almost all greedoid polynomials of rooted digraphs GM-factorise, or the ratio will approach a fixed α where $0.8299 \leq \alpha < 1$. We ask, what is the limiting proportion of greedoid polynomials of rooted digraphs that GM-factorise, as $n \rightarrow \infty$?

We categorised these polynomials into two classes, according to whether they are polynomials of separable or non-separable digraphs. Some of these polynomials are polynomials of both separable and non-separable digraphs. The number of such polynomials is given in column 6 (COMM) in Table 5. One such example for digraphs of order three is shown in Figure 2, where the two digraphs have the same greedoid polynomial $(1 + t)(1 + z)$.

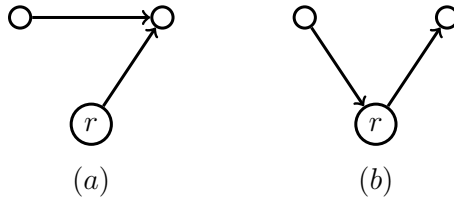


Figure 2: Digraphs that have the same greedoid polynomial where (a) is non-separable and (b) is separable

We are interested in non-separable digraphs that can be GM-factorised, especially those digraphs that have greedoid polynomials that are not the same as polynomials of any separable digraph. The numbers of greedoid polynomials of these digraphs are given in column PFNSU in Table 5, and examples of such rooted digraphs of order two and three are given in Figure 3 and Figure 4, respectively. It is easy to verify that the greedoid polynomial of the rooted digraph in Figure 3 is $(1 + t)(1 + z)$. The greedoid polynomials of rooted digraphs in Figure 4 are (from left to right starting from the first row) given in Table 6.



Figure 3: The non-separable digraph of order two that GM-factorises

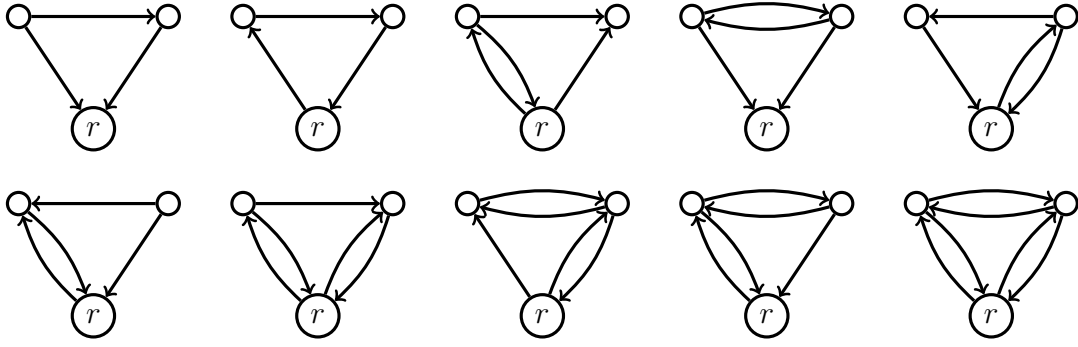


Figure 4: Ten of the 16 non-separable digraphs (one for each of the ten different greedoid polynomials) of order three that GM-factorise

Greedoid polynomials	Number of non-separable rooted digraphs of order three
1. $(1+z)^3$	2
2. $(1+z)(1+t+t^2+t^2z)$	3
3. $(1+z)(2+2t+t^2+z+tz+t^2z)$	2
4. $(1+z)^4$	1
5. $(1+z)^2(1+t+t^2+t^2z)$	3
6. $(1+t)(1+z)^3$	1
7. $(1+z)^2(2+2t+t^2+z+tz+t^2z)$	1
8. $(1+z)^2(3+2t+t^2+z+t^2z)$	1
9. $(1+z)^3(1+t+t^2+t^2z)$	1
10. $(1+z)^3(3+2t+t^2+z+t^2z)$	1

Table 6: Greedoid polynomials of non-separable digraphs of order three that GM-factorise and these polynomials are not the same as polynomials of any separable digraph of order three, and the numbers of associated non-separable digraphs (making 16 non-separable rooted digraphs altogether)

2.3 2-nonbasic and 1-primary Digraphs

We investigate greedoid polynomials that contain nonbasic and primary GM-factors. Details are given in Table 8 (see Table 7 for the list of abbreviations and Figure 5 for the corresponding Venn diagram). For rooted digraphs up to order six, each 1-primary digraph is a 2-nonbasic digraph, and each totally 1-primary digraph is a totally 2-nonbasic digraph.

Abbreviation	Description
2-NB	Number of 2-nonbasic digraphs
2-TNB	Number of totally 2-nonbasic digraphs
1-P	Number of 1-primary digraphs
1-TP	Number of totally 1-primary digraphs

Table 7: Abbreviations for Figure 5 and Table 8

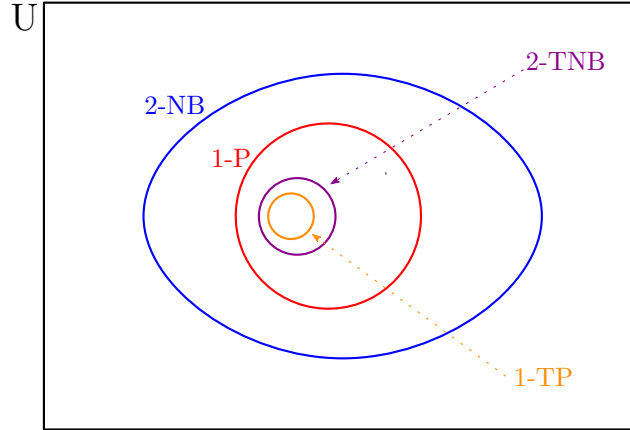


Figure 5: Venn diagram that represents four types of digraphs in Table 8 where U is the set of digraphs (up to order six) that can be GM-factorised

n	2-NB	2-TNB	1-P	1-TP
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	120	0	0	0
6	12348	15	1252	9

Table 8: Numbers of the four types of non-separable digraphs (up to order six) that can be GM-factorised

All rooted digraphs up to order four either have one nonbasic GM-factor or only basic GM-factors in their polynomials. There are 120 rooted digraphs of order five that have greedoid polynomials with at least two nonbasic GM-factors. The number of distinct greedoid polynomials of these 120 rooted digraphs is 34. Further examination showed that the number of nonbasic GM-factors in these polynomials is exactly two. Nonetheless, 117 of the 120 rooted digraphs have greedoid polynomials that contain at least one basic GM-factor, and the remaining three are separable digraphs (as shown in Figure 6).

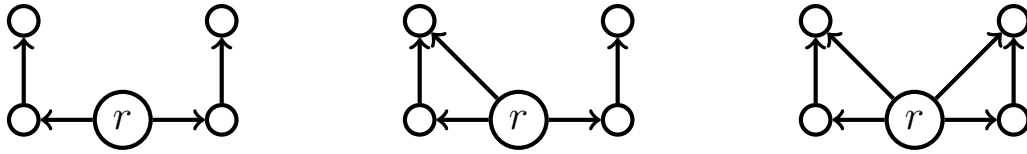


Figure 6: Three separable digraphs of order five that have two nonbasic GM-factors

Hence, there exist no totally 2-nonbasic digraphs of order five. In addition, none of the polynomials of these 120 rooted digraphs contains a primary GM-factor. This implies that none of the rooted digraphs up to order five are k -primary, for $k \geq 1$. Each of the GM-factors of greedoid polynomials of rooted digraph up to order five is either basic, or is a GM-factor of some greedoid polynomials of rooted digraphs of smaller order.

There are 12348 rooted digraphs of order six that have greedoid polynomials with at least two nonbasic GM-factors. The number of distinct greedoid polynomials of these 12348 rooted digraphs is 837. A quick search showed that all these digraphs are 2-nonbasic. We found that 15 of these rooted digraphs are totally 2-nonbasic. One of the totally 2-nonbasic digraphs D_1 of order six is shown in Figure 7 and its greedoid polynomial is as follows:

$$f(D_1) = (1 + t + t^2 + t^2z)(2 + 2t + t^2 + t^3 + z + tz + t^2z + 3t^3z + 3t^3z^2 + t^3z^3).$$

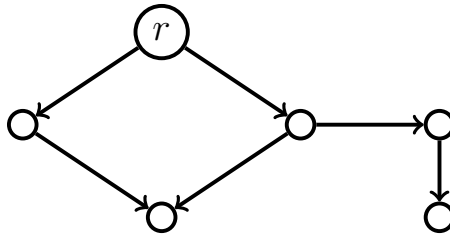


Figure 7: A totally 2-nonbasic digraph of order six

Both of the nonbasic GM-factors of $f(D_1)$ are greedoid polynomials of rooted digraphs G and H that have order three and four, respectively (see Figure 8). We have $f(G) = 1 + t + t^2 + t^2z$ and $f(H) = 2 + 2t + t^2 + t^3 + z + tz + t^2z + 3t^3z + 3t^3z^2 + t^3z^3$. However, D_1 is a non-separable digraph and hence not the direct sum of G and H .

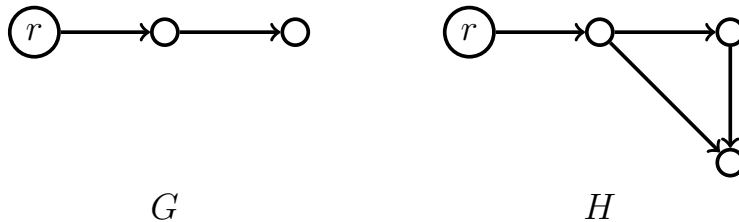


Figure 8: Rooted digraphs G and H

There are also 1252 rooted digraphs of order six that have greedoid polynomials with one primary GM-factor, and all these digraphs are non-separable. However, only nine of them are totally 1-primary digraphs. One of the totally 1-primary digraphs D_2 of order six is shown in Figure 9 and it has the following greedoid polynomial:

$$f(D_2) = (1+t+t^2+t^2z)(4+3t+t^2+t^3+4z+2tz+t^2z+4t^3z+z^2+6t^3z^2+4t^3z^3+t^3z^4).$$

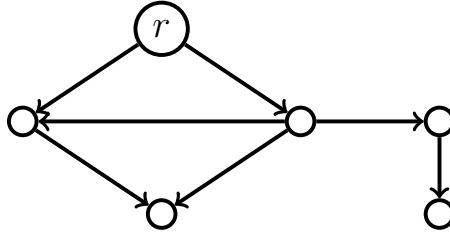


Figure 9: A totally 1-primary digraph of order six

The totally 1-primary digraph D_2 GM-factorises into one nonbasic GM-factor $1 + t + t^2 + t^2z$ and one primary GM-factor $4 + 3t + t^2 + t^3 + 4z + 2tz + t^2z + 4t^3z + z^2 + 6t^3z^2 + 4t^3z^3 + t^3z^4$. The GM-factor $1 + t + t^2 + t^2z$ is not primary as it is the greedoid polynomial of the rooted digraph G in Figure 8. Note that D_2 is also a totally 2-nonbasic digraph since every primary GM-factor is a nonbasic GM-factor.

The fact that a greedoid polynomial of a rooted digraph is not divisible by $1 + z$ implies that the associated rooted digraph has neither a directed cycle nor a greedoid loop. Our results show that there exist some non-separable digraphs (of order six) that GM-factorise into only nonbasic GM-factors, or both nonbasic and primary GM-factors. This implies that the multiplicative direct sum property, and the existence of greedoid loops and directed cycles, are not the only characteristics that determine if greedoid polynomials of rooted digraphs factorise.

2.4 An Infinite Family

Lastly, we show that there exists an infinite family of digraphs where their greedoid polynomials factorise into at least two nonbasic GM-factors. We first characterise greedoid polynomials of two classes of rooted digraphs.

Let P_{m,v_0} be a *directed path* $v_0v_1 \dots v_m$ of size $m \geq 0$ rooted at v_0 , and C_{m,v_0} be a *directed cycle* $v_0v_1 \dots v_{m-1}v_0$ of size $m \geq 1$ rooted at v_0 . For convenience, we usually write P_m for P_{m,v_0} and C_m for C_{m,v_0} .

Lemma 2.1.

$$f(P_m; t, z) = 1 + \frac{t(1 - (t(1+z))^m)}{1 - t(1+z)}.$$

Proof. By induction on the number of edges. □

Let G be a rooted undirected graph and $X \subseteq E(G)$. The *rank* $r(X)$ of X is defined as $r(X) = \max\{|A| : A \subseteq X, A \text{ is a rooted subtree}\}$. Let F be the set of subtrees of G containing the root vertex. Korte and Lovász [12] showed that (G, F) is a greedoid called the *undirected branching greedoid* of G .

Suppose Q_m is an *undirected path* $v_0v_1 \dots v_m$ of size $m \geq 0$ rooted at either v_0 or v_m . Then $f(P_m; t, z) = f(Q_m; t, z)$, since there is a rank-preserving bijection between $2^{E(P_m)}$ and $2^{E(Q_m)}$.

Lemma 2.2.

$$f(C_m; t, z) = (1 + z)f(P_{m-1}; t, z).$$

Proof. By induction on the number of edges. □

Gordon gave a formula for the greedoid polynomials of rooted undirected cycles in [7]. Those polynomials are different to the ones given by Lemma 2.2.

We now give an infinite family of digraphs where their greedoid polynomials factorise into at least two nonbasic GM-factors, extending the example in Figure 7.

Lemma 2.3. *There exists an infinite family of non-separable digraphs D that have at least two nonbasic GM-factors, where*

$$f(D) = f(P_{k+1}) (f(C_{k+1}) + f(P_{k+1}) + t^{k+2}(1+z)^{k+2}), \text{ for } k \geq 1.$$

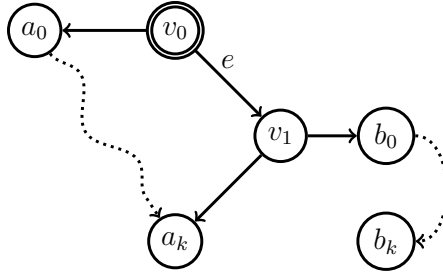


Figure 10: The digraph D in the proof of Lemma 2.3

Proof. Let D be the non-separable digraph rooted at vertex v_0 shown in Figure 10, where $a_0 \dots a_k$ and $b_0 \dots b_k$ are two directed paths in D of length $k \geq 1$ starting at a_0 and b_0 , respectively. To compute the greedoid polynomial of D by using Proposition 1.2, we first choose the edge $e = v_0v_1$. By deleting and contracting e , we obtain the digraphs $D_1 = D/e$ and $D_2 = D \setminus e$ as shown in Figure 11.

Note that D_1 is a separable digraph rooted at v_0 . Let $R = \{v_0, a_0, \dots, a_k\} \subset V(D_1)$, $S = \{v_0, b_0, \dots, b_k\} \subset V(D_1)$ and $T = \{v_0, a_0, \dots, a_k\} \subset V(D_2)$. Suppose $A = D_1[R]$ and $B = D_1[S]$ are the subdigraphs of D_1 induced by R and S respectively, and $C = D_2[T]$ is the subdigraph of D_2 induced by T . Clearly, $B \cong C \cong P_{k+1}$. Hence we have $f(B) = f(C) = f(P_{k+1})$. Note that every edge $g \in E(D_2) \setminus E(C)$ is a greedoid loop, and $|E(D_2) \setminus E(C)| = k + 2$. By using the recurrence formula, we have

$$f(D) = f(D/e) + t^{r(D)-r(D \setminus e)} f(D \setminus e)$$

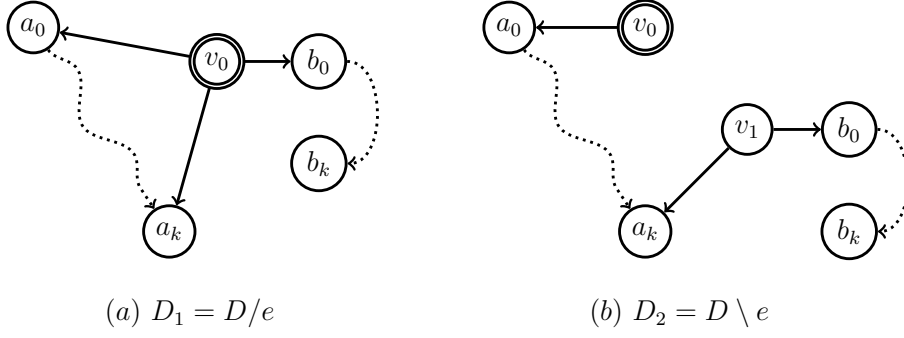


Figure 11: Two minors D/e and $D \setminus e$ of D

$$\begin{aligned}
&= f(A) \cdot f(B) + t^{(2k+3)-(k+1)} f(C) \cdot (1+z)^{k+2} \\
&= f(P_{k+1}) (f(A) + t^{k+2}(1+z)^{k+2}) \quad (\text{since } f(B) = f(C) = f(P_{k+1})).
\end{aligned}$$

It remains to show that $f(A)$ can be expressed in terms of $f(P_k)$ and $f(C_k)$. By taking $h = v_0 a_k \in E(A)$ (see Figure 12) as the outgoing edge in the recurrence formula, we have

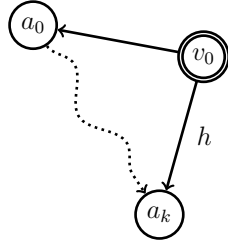


Figure 12: The subdigraph A of D_1 induced by R

$$\begin{aligned}
f(A) &= f(A/h) + t^{r(A)-r(A \setminus h)} f(A \setminus h) \\
&= f(C_{k+1}) + t^{(k+1)-(k+1)} f(P_{k+1}) \quad (\text{since } A/h \cong C_{k+1} \text{ and } A \setminus h \cong P_{k+1}) \\
&= f(C_{k+1}) + f(P_{k+1}).
\end{aligned}$$

Therefore,

$$f(D) = f(P_{k+1}) (f(C_{k+1}) + f(P_{k+1}) + t^{k+2}(1+z)^{k+2}).$$

Clearly, both factors of $f(D)$ are nonbasic GM-factors. Since D is non-separable and $k \geq 1$, the proof is complete. \square

We extend the infinite family in Lemma 2.3, and characterise the greedoid polynomials of a new infinite family, as follows.

Theorem 2.4. *There exists an infinite family of non-separable digraphs D that have at least two nonbasic GM-factors, where*

$$f(D) = f(P_{k+1}) \left(f(C_{k+1}) + f(P_{k+1}) + \frac{t^{k+2}(1+z)^{k+2}(1 - (t(1+z))^\ell)}{1 - t(1+z)} \right), \text{ for } k, \ell \geq 1.$$

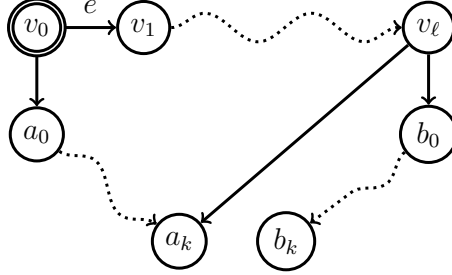


Figure 13: The digraph D in the proof of Theorem 2.4

Proof. Let D be the non-separable digraph rooted at vertex v_0 shown in Figure 13, where $L = v_0 \dots v_\ell$ is a directed path in D of length $\ell \geq 1$ starting at v_0 . We proceed by induction on the length ℓ of L .

For the base case, suppose $\ell = 1$. By Lemma 2.3, we have

$$\begin{aligned} f(D) &= f(P_{k+1}) \left(f(C_{k+1}) + f(P_{k+1}) + t^{k+2}(1+z)^{k+2} \right) \\ &= f(P_{k+1}) \left(f(C_{k+1}) + f(P_{k+1}) + \frac{t^{k+2}(1+z)^{k+2}(1 - (t(1+z))^\ell)}{1 - t(1+z)} \right), \end{aligned}$$

and the result for $\ell = 1$ follows.

Assume that $\ell > 1$ and the result holds for every $r < \ell$.

Let $e = v_0 v_1 \in E(D)$. By applying the deletion-contraction recurrence in Proposition 1.2 on e , we obtain the digraphs $D_1 = D/e$ and $D_2 = D \setminus e$ as shown in Figure 14.

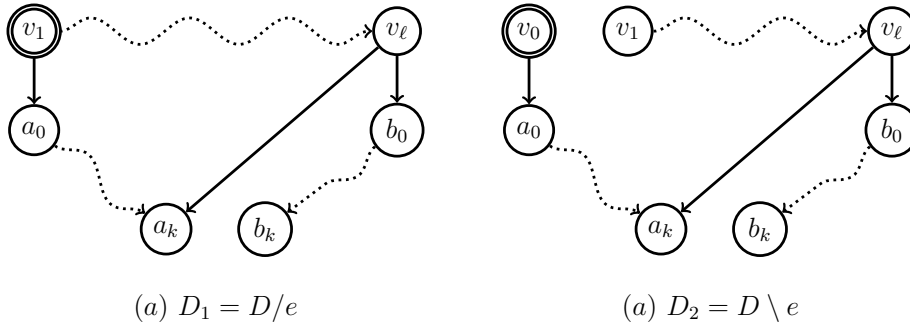


Figure 14: Two minors D/e and $D \setminus e$ of D

Note that D_1 is a non-separable digraph rooted at v_1 . Since the directed path $v_1 \dots v_\ell$ in D_1 has length $\ell - 1$, we use the inductive hypothesis to obtain $f(D_1)$. Let $R = \{v_0, a_0, \dots, a_k\} \subset V(D_2)$, and $A = D_2[R]$ be the subdigraph of D_2 induced by R . Clearly, $A \cong P_{k+1}$. Hence, we have $f(A) = f(P_{k+1})$. Note that every edge $g \in E(D_2) \setminus E(A)$ is a greedoid loop, and $|E(D_2) \setminus E(A)| = k + \ell + 1$. By using the recurrence formula, we have

$$\begin{aligned}
f(D) &= f(D/e) + t^{r(D)-r(D \setminus e)} f(D \setminus e) \\
&= f(P_{k+1}) \left(f(C_{k+1}) + f(P_{k+1}) + \frac{t^{k+2}(1+z)^{k+2}(1-(t(1+z))^{\ell-1})}{1-t(1+z)} \right) \\
&\quad + t^{(2k+\ell+2)-(k+1)} (f(P_{k+1}) \cdot (1+z)^{k+\ell+1}) \\
&= f(P_{k+1}) \left(f(C_{k+1}) + f(P_{k+1}) + \left(\frac{t^{k+2}(1+z)^{k+2}(1-(t(1+z))^{\ell-1})}{1-t(1+z)} \right) \right. \\
&\quad \left. + t^{k+\ell+1}(1+z)^{k+\ell+1} \right) \\
&= f(P_{k+1}) \left(f(C_{k+1}) + f(P_{k+1}) + \frac{t^{k+2}(1+z)^{k+2}(1-(t(1+z))^\ell)}{1-t(1+z)} \right).
\end{aligned}$$

□

We observe that if every directed path has length at most one in a digraph D rooted at a vertex v , the greedoid polynomial of D is trivial. In this scenario, every vertex in D is either a sink vertex or a source vertex. If v is a sink vertex, then every edge in D is a greedoid loop. If v is a source vertex, every edge that is not incident with v is a greedoid loop.

In the following theorem, we show that the greedoid polynomial of any digraph that has a directed path of length at least two is a nonbasic GM-factor of the greedoid polynomial of some non-separable digraph. The proof follows similar approaches as in Lemma 2.3 and Theorem 2.4.

Theorem 2.5. *For any digraph G that has a directed path of length at least two, there exists a non-separable digraph D where $f(D)$ has $f(G)$ as a nonbasic GM-factor.*

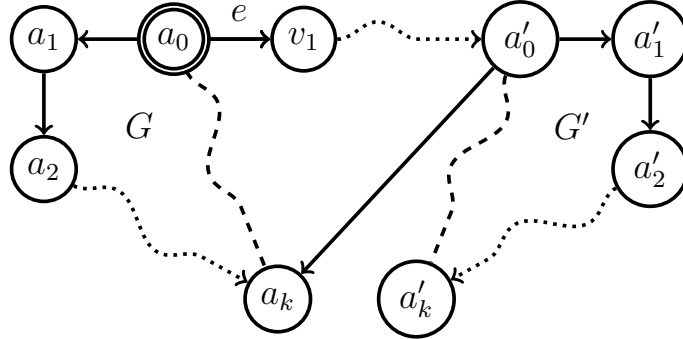


Figure 15: An illustration of the non-separable digraph D in Theorem 2.5

Proof. Let G be a digraph that has a directed path $K = a_0a_1 \dots a_k$ of length $k \geq 2$, and G' be a copy of G . The copy of K in G' is denoted by $K' = a'_0a'_1 \dots a'_k$.

We construct a non-separable digraph D_ℓ using G and G' , as follows. We first create a directed path $L = a_0v_1 \dots v_{\ell-1}a'_0$ of length ℓ . We add a directed edge a'_0a_k , and assign v_0 as the root vertex of D_ℓ (see Figure 15).

To show that $f(G)$ is a nonbasic GM-factor of $f(D_\ell)$, we proceed by induction on the length ℓ of L .

For the base case, suppose $\ell = 1$. We apply the deletion-contraction recurrence in Proposition 1.2 on $e = a_0a'_0$. We denote a_0 the root vertex of the separable digraph D_1/e . We have

$$\begin{aligned} f(D_1) &= f(D_1/e) + t^{r(D_1)-r(D_1 \setminus e)} f(D_1 \setminus e) \\ &= f(G + a_0a_k) \cdot f(G) + t^{(2r(G)+1)-r(G)} f(G) \cdot (1+z)^{|E(G)|+1} \\ &= f(G) (f(G + a_0a_k) + t^{r(G)+1} (1+z)^{|E(G)|+1}). \end{aligned}$$

Hence, the result for $\ell = 1$ follows.

Assume that $\ell > 1$ and the result holds for every $r < \ell$.

For the inductive steps, we apply the deletion-contraction recurrence on $e = v_0v_1$. We have

$$\begin{aligned} f(D_\ell) &= f(D_\ell/e) + t^{r(D_\ell)-r(D_\ell \setminus e)} f(D_\ell \setminus e) \\ &= f(D_\ell/e) + t^{(2r(G)+\ell)-r(G)} f(G) \cdot (1+z)^{|E(G)|+\ell} \\ &= f(D_\ell/e) + t^{r(G)+\ell} f(G) \cdot (1+z)^{|E(G)|+\ell}. \end{aligned}$$

Note that $D_\ell/e \cong D_{\ell-1}$. By the inductive hypothesis, $f(D_\ell/e)$ has $f(G)$ as a nonbasic GM-factor. This implies that $f(D_\ell)$ has $f(G)$ as a nonbasic GM-factor. \square

We now have the following corollary.

Corollary 2.6. *Let D be a non-separable digraph that belongs to the infinite family in Theorem 2.5. By replacing the edge $a'_0a_k \in E(D)$ by any digraph R such that every edge in $E(R)$ that is incident with a_k is an incoming edge of a_k , then $f(D)$ has $f(G)$ as a nonbasic GM-factor.* \square

3 Computational Methods

All labelled rooted digraphs (without loops and multiple edges, but cycles of size two are allowed) up to order six were provided by Brendan McKay on 28 March 2018 (personal communication from McKay to Farr). Each digraph is given as a list of numbers on one line separated by a single space. The first number is the order of the digraph, the second number is the size of the digraph, and each pair of subsequent numbers represent a directed edge of the digraph. For instance, 3 2 2 0 2 1 represents

a digraph of order 3 and size 2. The directed edges of the digraph are $(2, 0)$ and $(2, 1)$. Details are as follows:

$$\overbrace{3}^{\text{order}} \underbrace{2}_{\text{size}} \overbrace{2\ 0}^{\text{edge}} \underbrace{2\ 1}_{\text{edge}}.$$

We use a set of numbers $\{0, 1, \dots, n - 1\}$ to represent vertices for each digraph of order n , and an edge list to represent the edge set of each digraph, e.g., $[[0, 1]]$ represents a digraph with a single edge directed from vertex 0 to vertex 1. As there are 9,245,664 labelled rooted digraphs of order six, we split these digraphs into 52 files.

We use Python 3 (source code filename extension *.py*), Wolfram Mathematica 11 (source code filename extension *.nb*) and Bash Shell (Mac OS Version 10.13.4), in computing results for greedoid polynomials of rooted digraphs up to order six.

Algorithms of our programs are given in [Appendix](#). For brevity, we omit some elementary algorithms. Steps in obtaining our results are also summarised in [Appendix](#).

4 Concluding Remarks

In this paper, we presented (i) the results from exhaustive computation of all small rooted digraphs and (ii) the first results of the GM-factorability of greedoid polynomials of rooted digraphs.

We computed the greedoid polynomials for all rooted digraphs up to order six. From [Table 3](#), the ratio of PU to T-ISO shows a decreasing trend. We expect that as n increases, this ratio continues to decrease. Hence, we have the following conjecture.

Conjecture 4.1. *Most rooted digraphs are not determined by their greedoid polynomials.*

We found that the multiplicative direct sum property, and the existence of greedoid loops and directed cycles, are not the only characteristics that determine if greedoid polynomials of rooted digraphs factorise. We showed that there exists an infinite family of non-separable digraphs where their greedoid polynomials GM-factorise. We also characterised the greedoid polynomials of rooted digraphs that belong to the family.

We now suggest some problems for further research.

1. Investigate the factorability of greedoid polynomials of rooted graphs, or even greedoids in general.

Gordon and McMahon gave a graph-theoretic interpretation for the highest power of $1 + z$ for greedoid polynomials of rooted digraphs. We could investigate a similar problem for the other basic factor $1 + t$.

2. Does there exist a graph-theoretic interpretation for the highest power of $1 + t$ for greedoid polynomials of rooted digraphs?

By Theorem 2.5, we can see that there exist (totally) k -nonbasic rooted digraphs for $k \geq 3$.

3. For $k \geq 2$, does there exist a (totally) k -primary rooted digraph?

For rooted digraphs of order six, there are 15 totally 2-nonbasic digraphs and nine totally 1-primary digraphs.

4. For $k \geq 1$, can we characterise greedoid polynomials of totally $(k + 1)$ -nonbasic digraphs and totally k -primary digraphs?

5. Determine necessary and sufficient conditions for greedoid polynomials of rooted digraphs to factorise.

Acknowledgement

We thank Gary Gordon for his useful feedback.

References

- [1] A. Björner and G. M. Ziegler, Introduction to greedoids, *Matroid applications, Encyclopedia Math. Appl.*, Cambridge University Press, Cambridge, **40** (1992), 284–357.
- [2] B. Bollobás and L. Pebody and O. Riordan, Contraction-deletion invariants for graphs, *J. Combin. Theory Ser. B*, **80** (2000), 320–345.
- [3] T. H. Brylawski and J. Oxley, The Tutte polynomial and its applications, *Matroid applications, Encyclopedia Math. Appl.*, Cambridge University Press, Cambridge, **40**, (1992), 123–225.
- [4] T. Y. Chow, Digraph analogues of the Tutte polynomial, in: J. Ellis-Monaghan and I. Moffatt (eds.), *Handbook on the Tutte polynomial and related topics*, CRC Press, to appear.
- [5] F. R. K. Chung and R. L. Graham, On the cover polynomial of a digraph, *J. Combin. Theory Ser. B*, **65** (1995), 273–290.
- [6] D. Eisenstat and G. P. Gordon, Non-isomorphic caterpillars with identical subtree data, *Discrete Math.*, **306** (2006), 827–830.
- [7] G. P. Gordon, Chromatic and Tutte polynomials for graphs, rooted graphs, and trees, *Graph Theory Notes N. Y.*, **54** (2008), 34–45.
- [8] G. P. Gordon and E. W. McMahon, A greedoid polynomial which distinguishes rooted arborescences, *Proc. Amer. Math. Soc.*, **107** (2) (1989), 287–298.

- [9] G. P. Gordon and E. W. McMahon, Interval partitions and activities for the greedoid Tutte polynomial, *Advances in Applied Math.*, **18** (1997), 33–49.
- [10] A. A. Hagberg, D. A. Schult and P. J. Swart, Exploring network structure, dynamics, and function using NetworkX, in: *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11-15, Aug 2008.
- [11] B. Korte and L. Lovász, Mathematical structures underlying greedy algorithms, *Fundamentals of Computation Theory (Szeged, August 24–28, 1981)*, Lecture Notes in Comput. Sci., Springer, Berlin-New York, **117** (1981), 205–209.
- [12] B. Korte and L. Lovász, Greedoids - A structural framework for the greedy algorithm, in: *Progress in Combinatorial Optimization*, (1984), 221–243.
- [13] B. Korte and L. Lovász, Polymatroid greedoids, *J. Combin. Theory Ser. B*, **38** (1985), 41–72.
- [14] B. Korte, L. Lovász and R. Schrader, *Greedoids*, Springer, Berlin, 1991.
- [15] E. W. McMahon, On the greedoid polynomial for rooted graphs and rooted digraphs, *J. Graph Theory*, **17** (3) (1993), 433–442.
- [16] C. Merino, A. de Mier and M. Noy, Irreducibility of the Tutte polynomial of a connected matroid, *J. Combin. Theory Ser. B*, **83** (2) (2001), 298–304.
- [17] K. J. Morgan and G. E. Farr, Certificates of factorisation of chromatic polynomials, *Electron. J. Combin.*, **16** (2009), #R74.
- [18] S. J. Tedford, A Tutte polynomial which distinguishes rooted unicyclic graphs, *European J. Combin.*, **30** (2009), 555–569.
- [19] W. T. Tutte, A contribution to the theory of chromatic polynomials, *Can. J. Math.*, **6** (1954), 80–91.
- [20] W. T. Tutte, Duality and trinity, in: *Infinite and Finite Sets (Colloq., Keszthely, 1973)*, Vol. III, Colloq. Math. Soc. Janos Bolyai, Vol. 10, North-Holland, Amsterdam, 1973, pp. 1459–1475.
- [21] H. Whitney, A logical expansion in mathematics, *Bull. Amer. Math. Soc.*, **38** (1932), 572–579.

Appendix

We summarised commands and algorithms of our programs, as follows:

Part A

The relationships between files and programs in Part A is shown in Figure 16.

1. Program name: *Greedoid_polynomial.py* (see Algorithms 1–9)
Input: *dig_n.txt* that contains all digraphs of order n .
Output:
 - *dig_n_edgeList.txt*: contains edge lists for each rooted digraph of order n .
 - *dig_n_poly.txt*: contains greedoid polynomials (not in their simplest form) for each rooted digraph of order n which are obtained by using the deletion-contraction recurrence in Proposition 1.2.
 - *dig_n_isomorphism.txt*: contains rooted digraphs of order n together with a set of root vertices such that each digraph that has its root vertex in the set is isomorphic to each other.
 - *dig_n_directSum.txt*: summarises whether each rooted digraph of order n is a direct sum (DS), not a direct sum (NDS), or is isomorphic (ISO) to some other rooted digraphs.
 - *dig_n_info.txt*: contains the number of rooted digraphs of order n that are direct sums, and the number of rooted digraphs of order n that need to be excluded so that all rooted digraphs of order n are non-isomorphic.
2. Program name: *dig_n_factorise.nb*
Input: *dig_n_poly.txt*.
Output: *dig_n_poly_factorised.txt* that contains the greedoid polynomials for rooted digraphs of order n in their factorised forms.
3. Program name: *Numbering_edgeList.py*
Input: *dig_n_edgeList.txt*.
Output: *dig_n_edgeList_Numbering.txt* that includes the following numbering scheme for each line in the input file
$$n.z) \textit{edgeList}$$
where n is the order of the digraph and $z \geq 1$.
4. Program name: *Numbering_block_poly_factorised.py*
Input: *dig_n_poly_factorised.txt*.
Output: *dig_n_poly_factorised_Numbering.txt* that includes the following numbering scheme for each line in the input file
$$n.z.r) \textit{poly_factorised}$$
where n is the order of the digraph, $z \geq 1$ corresponds to the z^{th} rooted digraph

in the *dig_n_edgeList_Numbering.txt*, and $0 \leq r \leq n - 1$ represents the root vertex of the digraph.

5. Program name: *Numbering_block_directSum.py*

Input: *dig_n_directSum.txt*.

Output: *dig_n_directSum_Numbering.txt* that includes the following numbering scheme for each line in the input file:

$$n.z.r) DS/NDS/ISO$$

where n is the order of the digraph, $z \geq 1$ corresponds to the z^{th} rooted digraph in the *dig_n_edgeList_Numbering.txt*, and $0 \leq r \leq n - 1$ represents the root vertex of the digraph.

6. Program name: *DirectSum_vs_notDirectSum.py* (see Algorithm 10)

Input: *dig_n_directSum_Numbering.txt* and *dig_n_poly_factorised_Numbering.txt*.

Output:

- *dig_n_poly_directSum.txt*: contains greedoid polynomials for rooted digraphs of order n that are direct sums, with the respective numbering.
- *dig_n_poly_notDirectSum.txt*: contains greedoid polynomials for rooted digraphs of order n that are not direct sums, with the respective numbering.

7. Program name: *DirectSum_vs_notDirectSum_unique.py* (similar to Algorithm 10)

Input: *dig_n_directSum.txt* and *dig_n_poly_factorised.txt*.

Output:

- *dig_n_poly_directSum_unique.txt*: contains unique greedoid polynomials for rooted digraphs of order n that are direct sums.
- *dig_n_poly_notDirectSum_unique.txt*: contains unique greedoid polynomials for rooted digraphs of order n that are not direct sums.

8. Language: Bash Shell.

Input: *dig_n_poly_factorised.txt*.

Output: *dig_n_unique_poly.txt* that contains all unique greedoid polynomials of rooted digraphs of order n .

Command:

```
tr -d "\r" < dig_n_poly_factorised.txt | sort | uniq
> dig_n_unique_poly.txt
```

Remark: If we replace *uniq* by *uniq -c* in the above command, the last line of *dig_n_unique_poly.txt* gives the number of occurrences of “isomorphic”, which is also the number of rooted digraphs of order n that need to be excluded so that all rooted digraphs of order n are non-isomorphic. This number should match with the one in *dig_n_info.txt*.

9. Language: Bash Shell.

Input: *Combined_unique_poly_n-1.txt* and *dig_n_unique_poly.txt*.

Output: *Combined_unique_poly_n.txt* that contains all unique greedoid polynomials of rooted digraphs up to order n .

Command:

```
cat Combined_unique_poly_n-1.txt dig_n_unique_poly.txt | sort | uniq  
> Combined_unique_poly_n.txt
```

Remark: Since *Combined_unique_poly_1.txt* is literally *dig_1_unique_poly.txt*, we first use *dig_1_unique_poly.txt* and *dig_2_unique_poly.txt* as input files to obtain *Combined_unique_poly_2.txt*. For brevity, we only show *dig_n_unique_poly.txt* as the input in Figure 16. A similar concept is used for both steps 13 and 18.

10. Program name: *Factorability_unique.py* (see Algorithm 11)

Input: *dig_n_unique_poly.txt* and *Combined_unique_poly_n.txt*.

Output: *dig_n_factorability_unique.txt* that contains the number of greedoid polynomials of rooted digraphs of order n that can be GM-factorised, and output each of these polynomials.

11. Program name: *Factorability_unique_directSum.py* (similar to Algorithm 11)

Input: *dig_n_poly_directSum_unique.txt* and *Combined_unique_poly_n.txt*.

Output: *dig_n_factorability_directSum_unique.txt* that contains the number of greedoid polynomials that can be GM-factorised for rooted digraphs of order n that are direct sums, and output each of these polynomials.

12. Program name: *Factorability_unique_notDirectSum.py* (similar to Algorithm 11)

Input: *dig_n_poly_notDirectSum_unique.txt* and *Combined_unique_poly_n.txt*.

Output: *dig_n_factorability_notDirectSum_unique.txt* that contains the number of greedoid polynomials that can be GM-factorised for rooted digraphs of order n that are not direct sums, and output each of these polynomials.

13. Language: Bash Shell.

Input: *Combined_poly_directSum_n-1.txt* and *dig_n_poly_directSum.txt*.

Output: *Combined_poly_directSum_n.txt* that contains all unique greedoid polynomials of rooted digraphs that are direct sums up to order n .

Command:

```
cat Combined_poly_directSum_n-1.txt dig_n_poly_directSum.txt | sort  
| uniq > Combined_poly_directSum_n.txt
```

14. Program name: *DirectSum_and_GM-equivalent.py* (see Algorithm 12)

Input: *dig_n_poly_notDirectSum.txt* and *Combined_poly_directSum_n.txt*.

Output: *dig_n_poly_ndsEquivalent.txt* that contains the number of rooted digraphs of order n that are not direct sums, but they are GM-equivalent to some rooted digraph up to order n that is a direct sum. Each such polynomial will be printed out without duplicates in the output file.

15. Language: Bash Shell.
 Input: *dig_n_poly_factorised.txt*.
 Output: *dig_n_nonbasic.txt* that contains all unique polynomials that have at least two nonbasic GM-factors (these nonbasic GM-factors might be identical) for rooted digraphs of order n .
 Command:

```
sed -e 's/(1 + t)[^()*/; s/(1 + z)[^()*/' dig_n_poly_factorised.txt
| sort | uniq | grep ")\\*(\\|)^\\d\\+\\|([^+]* + [^+]*)"
> dig_n_nonbasic.txt
```
16. Language: Bash Shell.
 Input: *dig_n_nonbasic.txt*.
 Output: *dig_n_nonbasic_split.txt* that contains all unique nonbasic GM-factors that are split into separate lines for rooted digraphs of order n .
 Command:

```
sed -e 's/\\*(/'$'\n/g; s/(//; s/).*//' dig_n_nonbasic.txt
| tr -d "\\r" | sort | uniq > dig_n_nonbasic_split.txt
```
17. Language: Bash Shell.
 Input: *dig_n_poly_factorised.txt*.
 Output: *dig_n_factors.txt* that contains all unique GM-factors for greedoid polynomials of rooted digraphs of order n .
 Command:

```
sed -e 's/\\*(/'$'\n/g; /isomorphic/d; s/(//; s/).*//'
dig_n_poly_factorised.txt | tr -d "\\r" | sort | uniq
> dig_n_factors.txt
```
18. Language: Bash Shell.
 Input: *all_factors_up_to_order_n-1.txt* and *dig_n_factors.txt*.
 Output: *all_factors_up_to_order_n.txt* that contains all unique GM-factors for greedoid polynomials of rooted digraphs up to order n .
 Command:

```
cat all_factors_up_to_order_n-1.txt dig_n_factors.txt | sort | uniq
> all_factors_up_to_order_n.txt
```
19. Language: Bash Shell.
 Input: *dig_n_nonbasic_split.txt* and *all_factors_up_to_order_n-1.txt*.
 Output: *dig_n_primary.txt* that contains all primary GM-factors for greedoid polynomials of rooted digraphs of order n .
 Command:

```
comm -23 dig_n_nonbasic_split.txt all_factors_up_to_order_n-1
> dig_n_primary.txt
```
20. Language: Bash Shell.
 Input: *dig_n_primary.txt* and *dig_n_poly_factorised_Numbering.txt*.

Output: *dig_n_primaryPoly.txt* that contains all greedoid polynomials that have primary GM-factors for rooted digraphs of order n .

Command:

```
fgrep -f dig_n_primary.txt dig_n_poly_factorised_Numbering.txt  
> dig_n_primaryPoly.txt
```

21. Language: Bash Shell.

Input: *dig_n_primaryPoly.txt*.

Output: *dig_n_primaryNumbers_edgeList.txt* that contains the (edge list) numberings for all rooted digraphs of order n that have primary GM-factors in their greedoid polynomials.

Command:

```
awk '{print "^"$1}' dig_n_primaryPoly.txt | sed '/s/...$/)/'  
> dig_n_primaryNumbers_edgeList.txt
```

22. Language: Bash Shell.

Input: *dig_n_primaryPoly.txt*.

Output: *dig_n_primaryNumbers.txt* that contains the numberings for all greedoid polynomials that have primary GM-factors for rooted digraphs of order n .

Command:

```
awk '{print "^"$1}' dig_n_primaryPoly.txt > dig_n_primaryNumbers.txt
```

23. Language: Bash Shell.

Input: *dig_n_primaryNumbers_edgeList.txt* and *dig_n_edgeList_Numbering.txt*

Output: *dig_n_primaryGraphs.txt* that contains all edge lists for greedoid polynomials that have primary GM-factors for rooted digraphs of order n .

Command:

```
grep -f dig_n_primaryNumbers_edgeList.txt dig_n_edgeList_Numbering.txt  
> dig_n_primaryGraphs.txt
```

24. Language: Bash Shell.

Input: *dig_n_primaryNumbers.txt* and *dig_n_directSum_Numbering.txt*

Output: *dig_n_primaryVSdirectSum.txt* summarises whether each rooted digraph in *dig_n_primaryNumbers.txt* a direct sum or not a direct sum.

Command:

```
grep -f dig_n_primaryNumbers.txt dig_n_directSum_Numbering.txt  
> dig_n_primaryVSdirectSum.txt
```

25. Language: Bash Shell.

Input: *dig_n_primaryVSdirectSum.txt*

Output: *dig_n_primaryVSdirectSum_summary.txt* that contains rooted digraphs that are direct sums in *dig_n_primaryVSdirectSum.txt*.

Command:

```
grep "\tDS" dig_n_primaryVSdirectSum.txt  
> dig_n_primaryVSdirectSum_summary.txt
```

Part B

From Part A, we know that each rooted digraph (up to order six) that has $k \geq 1$ primary GM-factor in its greedoid polynomial is not a direct sum. We can now compute the number of greedoid polynomials of these digraphs that are not divisible by $1 + t$ or $1 + z$, which is also the number of totally k -primary digraphs.

26. Language: Bash Shell.

Input: *dig_n_primaryPoly.txt*

Output: *dig_n_totally_primaryPoly.txt* that contains greedoid polynomials that have primary GM-factors and are not divisible by $1 + t$ or $1 + z$ for rooted digraphs of order n that are not direct sums.

Command:

```
grep -v "(1 + t)\|(1 + z)" dig_n_primaryPoly.txt  
> dig_n_totally_primaryPoly.txt
```

By using a similar method, we compute the number of greedoid polynomials that contain at least two nonbasic GM-factors and are not divisible by $1 + t$ or $1 + z$, for rooted digraphs up to order six that are not direct sums.

27. Language: Bash Shell.

Input: *dig_n_nonbasic.txt*, *dig_n_poly_factorised_Numbering.txt* and *dig_n_directSum_Numbering.txt*

Output: *dig_n_totally_nonbasicPoly.txt* that contains greedoid polynomials that contain at least two nonbasic GM-factors and are not divisible by $1 + t$ or $1 + z$ for rooted digraphs of order n that are not direct sums.

Command:

```
fgrep -f dig_n_nonbasic.txt dig_n_poly_factorised_Numbering.txt |  
grep -v "(1 + t)\|(1 + z)" | awk '{print "^"$1}' | grep -f /dev/stdin  
dig_n_directSum_Numbering.txt | grep "\tNDS"  
> dig_n_totally_nonbasicPoly.txt
```

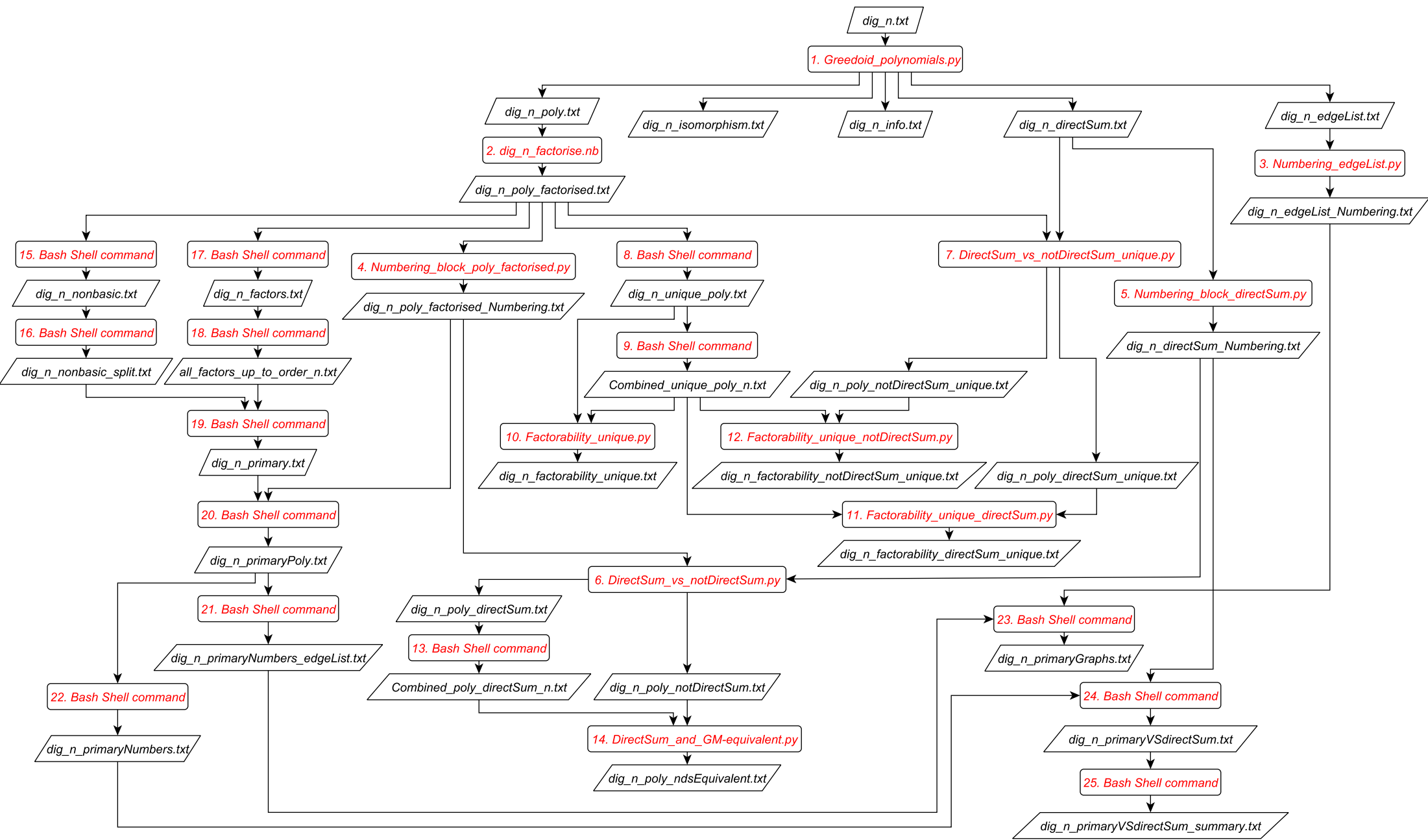


Figure 16: Relationships between files and programs

Algorithm 1 GreedoidPolynomial

Input: *dig_n.txt*

Output: *dig_n_edgeList.txt*, *dig_n_poly.txt*, *dig_n_isomorphism.txt*, *dig_n_info.txt* and *dig_n_directSum.txt*

```
1: digraphFile  $\leftarrow$  open the input file
2: edgeListTable  $\leftarrow$  create an empty list
3: for line in digraphFile do
4:   order  $\leftarrow$  first number of line
5:   aList  $\leftarrow$  create a list that excludes the first two numbers in line
6:   edgeList  $\leftarrow$  create a list of lists that has  $\lfloor aList/2 \rfloor$  empty lists
7:   for i in  $\lfloor aList/2 \rfloor$  do
8:     j  $\leftarrow 2i$ 
9:     Append aList[j] followed by aList[j + 1] to edgeList[i]
10:  Append edgeList to edgeListTable
11: Create output files: dig_n_edgeList.txt, dig_n_poly.txt, dig_n_isomorphism.txt,
    dig_n_info.txt and dig_n_directSum.txt
12: vertexList  $\leftarrow$  generate a vertex list numbered from 0 to order - 1
13: a  $\leftarrow 0$ 
14: k  $\leftarrow 0$ 
15: for item in edgeListTable do
16:   Write item to dig_n_edgeList.txt
17:   exclude  $\leftarrow$  create an empty list
18:   isomorphicTable  $\leftarrow$  IsomorphismTest(item,vertexList)
19:   if isomorphicTable is not empty then
20:     for element in isomorphicTable do
21:        $\ell$   $\leftarrow$  length of element
22:       k  $\leftarrow k + \ell - 1$ 
23:       Write item and element to dig_n_isomorphism.txt
24:       for node in element do
25:         Append each node except the first to exclude
26:       cutVertexList  $\leftarrow$  CutVertices(item)
27:       for node in exclude do
28:         if node in cutVertexList then
29:           Remove node from cutVertexList
30:       c  $\leftarrow$  length of cutVertexList
31:       a  $\leftarrow a + c$ 
32:       for vertex in vertexList do
33:         if vertex in cutVertexList then
34:           Write 'DS' to dig_n_directSum.txt
35:         else if vertex in exclude then
36:           Write 'ISO' to dig_n_directSum.txt
37:         else
38:           Write 'NDS' to dig_n_directSum.txt
```

Algorithm 2 GreedoidPolynomial (Cont.)

```
39:   for vertex in vertexList do
40:     if vertex in exclude then
41:       Write isomorphic to dig_n_poly.txt
42:     else
43:       Write DeletionContraction(vertexList,item,vertex) to dig_n_poly.txt
44: Write both a and k to dig_n_info.txt
```

Algorithm 3 IsomorphismTest(*edgeList,vertexList*)

Input: An edge list and a vertex list of a digraph

Output: Return a table where each list in the table contains vertices of the digraph in which, when vertices in the list are assigned as the root vertex of the digraph, these rooted digraphs are isomorphic to each other

```
1: rootList_table  $\leftarrow$  create an empty list
2: isomorphic  $\leftarrow$  create an empty list
3: checked  $\leftarrow$  create an empty list
4: for v1 in vertexList do
5:   if v1 is not in checked then
6:     Append v1 to checked
7:     rootList  $\leftarrow$  create a list that contains v1
8:     vertexList_new  $\leftarrow$  create a list that excludes the first element up to v1 in
       vertexList
9:     for v2 in vertexList_new do
10:      if Isomorphism(edgeList,v1,v2) is True then
11:        Append v2 to both rootList and checked
12:     if length of rootList > 1 then
13:       Append rootList to rootList_table
14:       temp  $\leftarrow$  0
15:       for element in rootList_table do
16:         if rootList is a subset of element then
17:           temp  $\leftarrow$  temp + 1
18:       if temp = 1 then
19:         Append rootList to isomorphic
20: return isomorphic
```

Algorithm 4 Isomorphism(*edgeList*,*r1*,*r2*)

Input: An edge list and two vertices of a digraph

Output: Return *True* if the digraph rooted at *r1* is isomorphic to the digraph rooted at *r2*, *False* otherwise

- 1: Import NetworkX package [10]
 - 2: $G1 \leftarrow$ append a loop incident with *r1* in the digraph
 - 3: $G2 \leftarrow$ append a loop incident with *r2* in the digraph
 - 4: **return** *nx.is_isomorphic*($G1, G2$)
-

Algorithm 5 CutVertices(*edgeList*)

Input: An edge list of a digraph

Output: Return a list of cutvertices of the digraph

- 1: Import NetworkX package [10]
 - 2: $G \leftarrow$ create an undirected multigraph using *edgeList*
 - 3: **return** *nx.articulation_points*(G)
-

Algorithm 6 DeletionContraction(*vertexList*,*edgeList*,*root*)

Input: A vertex list, an edge list and a root vertex of a digraph

Output: Return the greedoid polynomial of the digraph

- 1: **if** length of *edgeList* = 0 **then**
 - 2: **return** 1
 - 3: **else if** *Outdegree*(*edgeList*,*root*) = 0 **then**
 - 4: $r \leftarrow$ length of *edgeList*
 - 5: **return** $(1 + z)^r$
 - 6: **else**
 - 7: *edgeList_del* \leftarrow create a copy of *edgeList*
 - 8: *edgeList_con* \leftarrow create a copy of *edgeList*
 - 9: *vertexList_con* \leftarrow create a copy of *vertexList*
 - 10: *feasbile* \leftarrow *FeasibleSet.SizeOne*(*edgeList*,*root*)
 - 11: *randomEdge* \leftarrow choose a random edge from *feasbile*
 - 12: *edgeList_del* \leftarrow remove *randomEdge* from *edgeList_del*
 - 13: *contractedGraph* \leftarrow contract *randomEdge* in *edgeList_con*
 - 14: *edgeList_con* \leftarrow edge list of *contractedGraph*
 - 15: *vertexList_con* \leftarrow vertex list of *contractedGraph*
 - 16: *rank_ori* \leftarrow *RankFunction*(*vertexList*,*edgeList*,*root*)
 - 17: *rank_del* \leftarrow *RankFunction*(*vertexList*,*edgeList_del*,*root*)
 - 18: $k \leftarrow$ *rank_ori* - *rank_del*
 - 19: $d \leftarrow$ *DeletionContraction*(*vertexList*,*edgeList_del*,*root*)
 - 20: $c \leftarrow$ *DeletionContraction*(*vertexList_con*,*edgeList_con*,*root*)
 - 21: **return** $d * t^k + c$
-

Algorithm 7 Outdegree(edgeList,root)

Input: An edge list and a root vertex of a digraph

Output: Return the outdegree of the root vertex (loops are excluded)

```
1: outdegree  $\leftarrow$  0
2: for edge in edgeList do
3:   if the initial vertex of edge is root and the endvertex of edge is not root then
4:     outdegree  $\leftarrow$  outdegree + 1
5: return outdegree
```

Algorithm 8 FeasibleSet_SizeOne(edgeList,root)

Input: An edge list and a root vertex of a digraph

Output: Return the feasible set of size one of the digraph

```
1: feasible  $\leftarrow$  create an empty list
2: for edge in edgeList do
3:   if the initial vertex of edge is root and the endvertex of edge is not root then
4:     Append edge to feasible
5: return feasible
```

Algorithm 9 RankFunction(vertexList,edgeList,root)

Input: A vertex list, an edge list and a root vertex of a digraph

Output: Return the rank of the digraph

```
1: vertexList_new  $\leftarrow$  create a copy of vertexList
2: edgeList_new  $\leftarrow$  create a copy of edgeList
3: Remove root from vertexList_new
4: rootList  $\leftarrow$  create a list that contains root
5: k  $\leftarrow$  length of edgeList
6: for root in rootList do
7:   for edge in edgeList do
8:     if the initial vertex of edge is root and the endvertex of edge is in vertexList_new then
9:       Append the endvertex of edge to rootList
10:      Remove the endvertex of edge from vertexList_new
11:      Remove edge from edgeList_new
12: l  $\leftarrow$  length of edgeList_new
13: return k - l
```

Algorithm 10 DirectSum_vs_NotDirectSum

Input: *dig_n_directSum_Numbering.txt* and *dig_n_poly_factorised_Numbering.txt*

Output: *dig_n_poly_directSum.txt* and *dig_n_poly_notDirectSum.txt*

```
1: dsFile  $\leftarrow$  open dig_n_directSum_Numbering.txt
2: polyFile  $\leftarrow$  open dig_n_poly_factorised_Numbering.txt
3: dsPolyFile  $\leftarrow$  create an output file dig_n_poly_directSum.txt
4: ndsPolyFile  $\leftarrow$  create an output file dig_n_poly_notDirectSum.txt
5:  $k \leftarrow 1$ 
6: while  $k \leq$  number of lines in dsFile do
7:   if  $k^{\text{th}}$  line in dsFile contains 'DS' then
8:     Write the  $k^{\text{th}}$  line in polyList to dsPolyFile
9:   else if  $k^{\text{th}}$  line in dsFile contains 'NDS' then
10:    Write the  $k^{\text{th}}$  line in polyList to ndsPolyFile
11:     $k \leftarrow k + 1$ 
```

Algorithm 11 Factorability_Unique

Input: *dig_n_unique_poly.txt* and *Combined_unique_poly_n.txt*

Output: *dig_n_factorability_unique.txt*

```
1: polyFile  $\leftarrow$  open dig_n_unique_poly.txt
2: combinedFile  $\leftarrow$  open Combined_unique_poly_n.txt
3: factorabilityFile  $\leftarrow$  create an output file dig_n_factorability_unique.txt
4:  $k \leftarrow 0$ 
5: for oriPoly that has more than one factor in polyFile do
6:   for poly1 in combinedFile do
7:     for poly2 in combinedFile that excludes the first element up to the element
       before poly1 do
8:       if  $poly1 * poly2 = oriPoly$  then
9:         Write oriPoly to factorabilityFile
10:         $k \leftarrow k + 1$ 
11:        Break and move to the next element in polyFile
12: Write  $k$  to factorabilityFile
```

Algorithm 12 DirectSum_and_GM-equivalent

Input: *dig_n_poly_notDirectSum.txt* and *Combined_poly_directSum_n.txt*

Output: *dig_n_poly_ndsEquivalent.txt*

```
1: ndsPolyFile  $\leftarrow$  open dig_n_poly_notDirectSum.txt
2: combinedDsPolyFile  $\leftarrow$  open Combined_poly_directSum_n.txt
3: equivalentFile  $\leftarrow$  create an output file dig_n_poly_ndsEquivalent.txt
4:  $f \leftarrow 0$ 
5: for ndsPoly in ndsPolyFile do
6:   for dsPoly in combinedDsPolyFile do
7:     if second column of ndsPoly = second column of dsPoly then
8:       Write the second column of ndsPoly to equivalentFile
9:        $f \leftarrow f + 1$ 
10:    Break and move to the next element in ndsPolyFile
11: Write  $f$  to equivalentFile
```
