The Uniform Generalized Decomposition Method for Generating Algebraic Expressions of Fibonacci Graphs

MARK KORENBLIT and VADIM E. LEVIT
Department of Computer Science
Holon Academic Institute of Technology
52 Golomb Str., P.O. Box 305, Holon 58102
ISRAEL

Abstract: - The paper investigates relationship between algebraic expressions and Fibonacci graphs (which give a generic example of non-series-parallel graphs). We propose the uniform generalized decomposition method for constructing Fibonacci graph expressions. On every step this method divides the graph of size $n$ into $k$ parts of the same size. We prove that to reach the smallest possible length of the corresponding algebraic expression the number $k$ should be chosen equal to 2. When $k$ is 2, the method generates expressions of $O(n^2)$ length.

Key-Words: - complexity of an algebraic expression, Fibonacci graph, optimal representation of the algebraic expression, st-dag, st-dag expression, subexpression, subgraph

1 Introduction

A graph $G=(V,E)$ consists of a vertex set $V$ and an edge set $E$, where each edge corresponds to a pair $(v,w)$ of vertices. If the edges are ordered pairs of vertices (i.e., the pair $(v,w)$ is different from the pair $(w,v)$), then we call the graph directed or digraph; otherwise, we call it undirected. A path from vertex $v_0$ to vertex $v_j$ in a graph $G=(V,E)$ is a sequence of its vertices $v_0, v_1, v_2, \ldots, v_{j-1}, v_j$ such that $(v_i,v_{i+1}) \in E$ for $1 \leq i \leq j$. A graph $G'=(V',E')$ is a subgraph of $G=(V,E)$ if $V' \subseteq V$ and $E' \subseteq E$. A graph $G$ is homeomorphic to a graph $G'$ (homeomorph of $G'$) if $G$ can be obtained by subdividing edges of $G'$ via adding new vertices. A two-terminal directed acyclic graph (st-dag) has only one source $s$ and only one sink $t$. In an st-dag, every vertex lies on some path from the source to the sink.

An algebraic expression is called an st-dag expression if it is algebraically equivalent to the sum of products corresponding to all possible paths between the source and the sink of the st-dag [1]. This expression consists of terms (edge labels), and the operators $+$ (disjoint union) and $\cdot$ (concatenation, also denoted by juxtaposition when no ambiguity arises).

We define the complexity of an algebraic expression in two ways. The complexity of an algebraic expression is (i) a total number of terms in the expression including all their appearances (the first complexity characteristic) or (ii) a number of plus operators in the expression (the second complexity characteristic). The first and the second complexity characteristics of an st-dag expression are denoted by $T(n)$ and $P(n)$ respectively, where $n$ is the number of vertices of the graph (the size of the graph). An equivalent expression with the minimum complexity is called an optimal representation of the algebraic expression.

A series-parallel (SP) graph is defined recursively as follows:

(i) A single edge $(u,v)$ is a series-parallel graph with source $u$ and sink $v$.

(ii) If $G_1$ and $G_2$ are series-parallel graphs, so is the graph obtained by either of the following operations:

(a) Parallel composition: identify the source of $G_1$ with the source of $G_2$ and the sink of $G_1$ with the sink of $G_2$.

(b) Series composition: identify the sink of $G_1$ with the source of $G_2$.

As shown in [1] and [7], a SP graph expression has a representation in which each term appears only once. We proved in [7] that this representation is an optimal representation of the SP graph expression from the first complexity characteristic point of view. For example, the expression of the SP graph presented in Fig. 1 is $abd+abe+acd+ace+fe+fd$.

Fig.1. A series-parallel graph.
This expression can be reduced to \((a(b+c)+f)(d+e)\), where each term appears once.

The notion of a Fibonacci graph (FG) was introduced in [6]. In such an st-dag, two edges leave each of its \(n\) vertices except the two final vertices \((n-1)\) and \(n\). Two edges leaving the \(i\) vertex \((1 \leq i \leq n-2)\) enter the \(i+1\) and the \(i+2\) vertices. The single edge leaving the \(n-1\) vertex enters the \(n\) vertex. No edge leaves the \(n\) vertex. This graph is illustrated in Fig. 2.

As shown in [3], an st-dag is SP if and only if it does not contain a subgraph homeomorphic to the forbidden subgraph enclosed between vertices 1 and 4 of an FG from Fig. 2. Thus, Fibonacci graphs are of interest as "through" non-series-parallel st-dags. Notice that FGs of size 2 or 3 are series-parallel.

Mutual relations between graphs and algebraic expressions are discussed in [1], [7], [8], [9], [10], [11], [12], and [13], and other works. Specifically, [10], [11], and [13] consider the correspondence between SP graphs and read-once functions. A Boolean function is defined as read-once if it may be computed by some formula in which no variable occurs more than once (read-once formula). On the other hand, a SP graph expression can be reduced to the representation in which each term appears only once. Hence, such a representation of a SP graph expression can be interpreted as a read-once formula (Boolean operations are replaced by arithmetic ones).

An expression of a homeomorph of the forbidden subgraph belonging to any non-SP st-dag has no representation in which each term appears once. For example, consider the subgraph enclosed between vertices 1 and 4 of an FG from Fig. 2. Possible optimal representations of its expression are \(a_1(a_2a_3+b_2)+b_3\) or \((a_1a_3+b_1)a_4+a_1b_2\). For this reason, an expression of a non-SP st-dag cannot be represented as a read-once formula. However, for arbitrary functions, which are not read-once, generating the optimum factored form is NP-complete [14]. Some heuristic algorithms developed in order to obtain good factored forms are described in [4], [5], and other works. Therefore, generating an optimal representation (from both complexity characteristics point of view) for a non-SP st-dag expression is a highly complex problem.

Our intent in this paper is to simplify the expressions of Fibonacci graphs. In [9] we presented a heuristic algorithm based on a decomposition method with that end in view. The paper [7] is a substantially improved version of [9]. Here we adduce a more generalized method (the generalized decomposition method).

## 2 The Decomposition Method for Constructing Fibonacci Graph Expressions

Consider the \(n\)-vertex FG presented in Fig. 2. For brevity, we identify vertices by their ordinal numbers. Denote by \(E(p,q)\) a subexpression related to this subgraph (which is an FG also) having a source \(p\) \((1 \leq p < n)\) and a sink \(q\) \((1 < q \leq n, q > p)\). We choose any decomposition vertex \(i\) \((p < i < q)\) on a subgraph, and, in effect, split it at this vertex (Fig. 3). As follows from FG structure any path from vertex \(p\) to vertex \(q\) passes through vertex \(i\) or avoids it via edge \(b_{i+1}\). Therefore, \(E(p,q)\) is generated by the following recursive procedure (decomposition procedure):

1. **case** \(q = p\): \(E(p,q) \leftarrow 1\)
2. **case** \(q = p + 1\): \(E(p,q) \leftarrow a_p\)
3. **case** \(q > p + 1\): **choice**\((p,q,i)\)
4. \(E(p,q) \leftarrow E(p,i)E(i,q) + E(p,i-1)b_{i+1}E(i+1,q)\)

![Fig.2. An \(n\)-vertex Fibonacci graph.](image)

![Fig.3. Decomposition of a Fibonacci subgraph at vertex \(i\).](image)
The representation with a minimum total number of terms among all possible representations of Ex(FG) derived by the decomposition method is achieved if and only if on each recursive step i is equal to $\frac{q-p+2}{2}$ for odd q–p+1 and to $\frac{q-p+1}{2}$ or $\frac{q-p+3}{2}$ for even q–p+1, i.e., when i is a middle vertex of the interval (p,q). This decomposition method is called optimal.

It can be shown that the representation with a minimum number of plus operators among all possible representations of Ex(FG) derived by the decomposition method can be achieved also by the optimal decomposition method.

Thus, the total number of terms in the n-vertex FG expression derived by the optimal decomposition method is defined recursively as follows:

$$T(1) = 0, T(2) = 1$$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor + 1 \right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor - 1 \right) + T\left(\left\lceil \frac{n}{2} \right\rceil \right) + 1, n \geq 3.$$
and

\[ P(n) = \frac{1}{3} \left( \frac{7}{16} n^2 - 1 \right). \]

For example, for the 8-vertex FG, the possible expression derived by the optimal decomposition method is

\[(a_1(a_2a_3+b_1)+b_1a_3)((a_4a_5+b_2)(a_6a_7+b_6)+a_8a_9b_7)+
(a_4a_5+b_1)b_3(a_5a_6+b_6)+b_7a_9b_8).\]

It contains 25 terms and 9 plus operators.

We conjecture that the optimal decomposition method provides an optimal representation (for both our complexity characteristics) of an algebraic expression of a Fibonacci graph.

3 The Generalized Decomposition (GD) Method for Constructing Fibonacci Graph Expressions

As follows from the previous section, the decomposition method is based on splitting an FG on each recursive step into two parts via decomposition vertex i and edge b_{i-1}. The GD method proposes splitting an FG on each recursive step into an arbitrary number of parts (we will denote this number as k) via decomposition vertices \(i_1, i_2, \ldots, i_{k-1}\) and edges \(b_{i-1}, b_{i-2}, \ldots, b_{i_{k-1}}\) respectively. For instance, an example for \(k = 3\) is illustrated in Fig. 4.

In all cases when \(k > 2\), the decomposition procedure used in the previous section is transformed to the more complex form. Specifically, for \(k = 3\), the general line of the new decomposition procedure corresponding to line 4 of the previous decomposition procedure is presented as

\[ E(p,q) \leftarrow E(p,i_1)E(i_1,i_2)E(i_2,q)+
E(p,i_1-1)b_{i-1}E(i_1+1,i_2)E(i_2,q)+
E(p,i_2)b_{i-1}E(i_1,i_2-1)b_{i-1}E(i_2+1,q)+
E(p,i_2-1)b_{i-1}E(i_1+1,i_2-1)b_{i-1}E(i_2+1,q).\] (1)

The sum above consists of four parts, and each part includes three subexpressions corresponding to the three parts of a splitted subgraph. Hence, a current subgraph is decomposed into twelve new subgraphs.

In the general case, a number of subgraphs after a decomposition does not exceed \(2^{k-1}k\). The exact value of this number depends on the arrangement of decomposition vertices. For example, for a 4-vertex FG, \(k = 3\), the following expression should be generated formally, according to (1):

\[ a_1a_2a_3+b_1a_3+a_1b_2+b_1b_2.\]

However, there is no path \(b_1b_2\) in the graph (decomposition vertex 3 is adjacent to decomposition vertex 2 and, therefore, edges \(b_1\) and \(b_2\) cross but do not belong to the same path). Hence, the part \(b_1b_2\) of the expression is accepted to be equal to 0 and, finally, we have

\[ a_1a_2a_3+b_1a_3+a_1b_2.\]

We denote the number of additional edges (labeled b with index) which connect revealed subgraphs in an FG by \(T_b\). In such a case,

\[ T_b \leq \sum_{y=0}^{k-1} C^y_{k-1} \cdot y = 2^{k-2}(k-1).\]

The number of additional plus operators between parts of the derived expression is denoted by \(P_+\) and is estimated as

\[ P_+ \leq 2^{k-1} - 1.\]

Suppose that an FG is splitted into approximately equal parts on each recursive step (distances between decomposition vertices are equal or close). It will be the uniform GD method. In such a case, for an \(n\)-vertex FG, the total number of terms \(T(n)\) in the expression \(Ex(FG)\) derived by the uniform GD method can be estimated for large \(n\) as follows:

![Fig.4. Decomposition of a Fibonacci subgraph at vertices \(i_1\) and \(i_2\).](image)
\[ T(n) \leq 2^{k-1} kT \left( \frac{n}{k} \right) + 2^{k-2} (k-1). \]

By the master theorem
\[ T(n) = O\left(n^{\log_k (2^{k-1})}\right) = O(n^{1+\log_k 2^{k-1}}). \]

Analogously, for an \( n \)-vertex \( FG \), the number of plus operators \( P(n) \) in the expression \( Ex(FG) \) derived by the uniform GD method can be estimated for a large \( n \) as follows:

\[ P(n) \leq 2^{k-1} kT \left( \frac{n}{k} \right) + 2^{k-1} - 1. \]

Hence, by the master theorem
\[ P(n) = O\left(n^{1+\log_k 2^{k-1}}\right). \]

As follows from (2) and (3) \( T(n) \) and \( P(n) \) reach the minimum complexity among \( 2 \leq k \leq n-1 \) when \( k = 2 \). Substituting 2 for \( k \) in (2) and (3) gives \( O(n^2) \) (we have the optimal decomposition method in such a case). Further, the complexity increases with increase of \( k \). For example, we have \( O\left(n^{1+\log_2 3}\right) \) for \( k = 3 \), \( O\left(n^{2.5}\right) \) for \( k = 4 \), etc. In the extreme case, when \( k = n-1 \), all inner vertices (from 2 to \( n-1 \)) of an \( n \)-vertex \( FG \) are decomposition vertices. The single recursive step is executed in such a situation, and all revealed subgraphs are individual edges (labeled \( a \) with index) connected by additional edges (labeled \( b \) with index). Hence, we have the sum of products corresponding to all possible paths between the source and the sink of the \( FG \). This sum is a base for the definition of the st-dag expression. The method generating such an \( FG \) expression representation is called the sequential paths method [9]. That is, the uniform GD method is reduced to the sequential paths method in such a case. Substituting \( n-1 \) for \( k \) in (2) and (3) gives
\[ O\left(n^{1+\log_k 2^{k-1}}\right) > O\left(n^{1+\log_k 2^{2-2}}\right) = O(2^{2-2} n). \]

The example, when the uniform GD method turns out to be the sequential paths method is adduced above, for the 4-vertex \( FG \). For the 8-vertex \( FG \), the corresponding algebraic expression is
\[ a_1a_2b_3a_4a_5a_6+\alpha a_1a_2b_3a_4a_5a_6+\alpha a_1a_2b_3a_4a_5a_6+\alpha a_1a_2b_3a_4a_5a_6+\alpha a_1a_2b_3a_4a_5a_6 \]

It contains 21 products (that correspond to 21 sequential paths of the graph), 109 terms and 20 plus operators. These results do not contradict to our conjecture that the optimal decomposition method provides an optimal representation. At least, it is the best one among uniform GD methods (asymptotically).

4 Conclusion and Future Work

The paper investigates relationship between algebraic expressions and Fibonacci graphs. The optimal decomposition method applied to an \( n \)-vertex \( FG \) generates an expression with \( O(n^3) \) complexity. This method is the best one among uniform generalized decomposition methods. In the extreme case the uniform generalized decomposition method is reduced to the sequential paths method that generates an expression of exponential complexity. We do not have a formal proof that splitting a \( FG \) into approximately equal parts gives the optimal result for an arbitrary number of parts (as in Theorem 1 where the number of parts equals two). Besides, the GD method considers splitting a \( FG \) into the same number of parts on each recursive step. One further generalization of the method assigns to any subgraph its own number of decomposition vertices. While, there exist representations derived by algorithms which can not be related even to generalized decomposition methods, we conjecture that the optimal decomposition method yields the shortest expression possible.

References:


