Telling Stories Fast
Via Linear-Time Delay Pitch Enumeration

Michele Borassi[1,2], Pierluigi Crescenzi[3], Vincent Lacroix[4], Andrea Marino[3], Marie-France Sagot[4], and Paulo Vieira Milreu[4]

[1] Scuola Normale Superiore, 56126 Pisa, Italy
[2] Università di Pisa, Dipartimento di Matematica, 56127 Pisa, Italy
[3] Università di Firenze, Dipartimento di Sistemi e Informatica, 50134 Firenze, Italy
[4] Inria Rhône-Alpes & Université de Lyon, F-69000 Lyon; Université Lyon 1; CNRS, UMR5558, Laboratoire de Biométrie et Biologie Évolutive, 69622 Villeurbanne, France

Abstract. This paper presents a linear-time delay algorithm for enumerating all directed acyclic subgraphs of a directed graph $G(V, E)$ that have their sources and targets included in two subsets S and T of V, respectively. From these subgraphs, called pitches, the maximal ones, called stories, may be extracted in a dramatically more efficient way in relation to a previous story telling algorithm. The improvement may even increase if a pruning technique is further applied that avoids generating many pitches which have no chance to lead to a story. We experimentally demonstrate these statements by making use of a quite large dataset of real metabolic pathways and networks.

1 Introduction

Directed graphs are a widely used model in computational biology, notably to represent metabolism, which is the set of chemical transformations that sustain life. If an organism is exposed to a given condition (for instance, some kind of stress), the vertices of the directed graph may be colored depending on whether the quantity of the chemical the vertex represents changed (one color, say black) or remained the same (another color, say white) in relation to what may be defined as the organism's "normal state". Data such as these may be obtained through a technique called metabolomics [9] whose need for analytical methods is giving rise to new research topics. One question of interest then is to understand which subparts of the graph are affected by the condition change. One biologically pertinent definition for such subparts is as follows [7]: a maximal directed acyclic subgraph whose sets of sources and targets are blacks (note that black vertices may also be internal, that is neither sources nor targets, but white vertices can only be internal). In [1], these subgraphs have been called *metabolic stories*, or *stories* for short. Stories are a novel object for the analysis of metabolomics data, but we believe that they may also be useful in other domains. In this paper, we are interested in efficiently enumerating all the stories included in a directed graph.

V. Bonifaci et al. (Eds.): SEA 2013, LNCS 7933, pp. 200–211, 2013.

Enumerating maximal directed acyclic subgraphs of a given directed graph G, without any constraint on their sources and targets, is equivalent to enumerating all feedback arc sets of G, which is itself a classical problem in computer science. An elegant polynomial-time delay algorithm for solving this problem was proposed by Schwikowski and Speckenmeyer [2]. In [1], however, it was shown that the constraint on the sets of sources and targets is enough to drastically change the nature of the problem. Although the complexity of enumerating stories remains open, in [1] the authors proposed an algorithm that is able to go to completion for small enough graphs, and that can be used in a randomized fashion in the case of larger graphs, in order to produce a large sample of stories (as far as we know, this is the only known algorithm for enumerating stories). This algorithm is based on the notion of *pitch*, which is defined as a story without the maximality constraint, and on the following fact: any permutation π of the vertices of G can be transformed in polynomial time to a pitch P_π so that, for any story $\mathcal{S}$, there exists a permutation π of the vertices in G such that P_π can be "completed" in polynomial time in order to obtain $\mathcal{S}$. The algorithm for enumerating stories then proceeds by enumerating all permutations, transforming each of them into the corresponding pitch, and completing this pitch into a story. Unfortunately, this algorithm, called Gobbolino, is not polynomial-time delay, that is, the time between the generation of two distinct stories can be exponential in the number of vertices of G (for definitions concerning enumeration algorithms and complexity we refer the reader to the seminal paper [3]). For this reason, in [1] a randomized implementation of the algorithm has been suggested, which simply generates permutations uniformly at random: a biological application of Gobbolino and of its randomized version is described in [6].

The main contribution of this paper is twofold. From a theoretical point of view, we show that pitches can be enumerated in linear-time delay. In particular, we show how pitches can be sorted in a rooted tree $\mathcal{T}$ and how a depth-first search of $\mathcal{T}$ can be performed while ensuring the linear-time delay constraint, by applying the so-called *reverse search* technique [4]. From a practical point of view, we propose a new algorithm for enumerating stories, called Touche, which is based on the linear-time delay pitch enumeration and on the pitch completion mechanism introduced in [1]. In particular, we first show how the depth-first search of $\mathcal{T}$ can be made more efficient by using a pruning technique which allows us to avoid visiting parts of the tree that certainly do not contain any story, and we then experimentally compare the Gobbolino algorithm with the Touche algorithm, on a large dataset of metabolic networks. Our experiments show that Touche always significantly outperforms Gobbolino, and that it is able to enumerate *all* stories in the case of bigger networks for which Gobbolino is not even able to produce a significant fraction of them.

1.1 Preliminaries

Let $G(V, E, S, T)$ be a directed graph, where V is the set of all vertices of G, E the set of arcs, and S and T two subsets of V. A vertex u is said to be a *source* if its out-degree is greater than 0 and its in-degree is 0, and it is said to be a

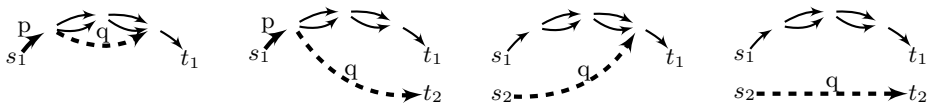

Fig. 1. The visualization of a pitch and a child obtained by adding the dashed path q. The path p is outlined when it is not empty.

target if its in-degree is greater than 0 and its out-degree is 0. A *pitch* P of G is a set of arcs $E' \subseteq E$, such that the subgraph $G' = (V', E')$ of G, where $V' \subseteq V$ is the set of vertices of G having at least one out-going or in-coming arc in E', is acyclic and for each vertex $w \in V' - S$, w is not a source in G', and for each vertex $w \in V' - T$, w is not a target in G'. We say that a pitch P is a story if it is maximal. A vertex w is said to belong to P if it belongs to V'. A path p is simple by definition and is denoted by $p_0, \dots, p_{|p|}$. We refer to a path p by its natural sequence of vertices or set of edges. We will assume without loss of generality that, for each vertex v, there is a path from a source to v and from v to a target (otherwise we may remove v from the graph). The vertices in S and T are said to be black, while the vertices in $(V - S) - T$ are said to be white. It is worth observing that, besides the fact that a pitch may contain a subset of the black vertices instead of all of them, we work in this paper with a generalization of the definition of pitch introduced in [1] in the sense that here, instead of considering one set of black vertices, we distinguish between black source vertices (they form the set S) and black target vertices (they form the set T). The problem treated in [1] corresponds to the case in which all black vertices are both in S and in T. Finally, we refer to $|V| + |E|$ as the size of the graph $|G|$.

2 Enumerating Pitches

In order to enumerate all the pitches contained in a graph $G = (V, E, S, T)$, we first sort them in a rooted tree $\mathcal{T}$, and then we perform a depth-first search of $\mathcal{T}$. In order to construct $\mathcal{T}$, we introduce an appropriately defined child relationship, such that, for every pitch Q, there exists one and only one pitch P such that Q is a child of P: P is said to be *the father* of Q, and it can be computed starting from Q via a linear-time computable function `father`. A child of a pitch P is always obtained by attaching to P a path q "outside P", that is, such that each internal vertex of q is outside P and each arc of q is not in P (see Fig. 1). We will also impose that in P there is a (possibly empty) path p such that each path in $P \cup q$ from a source to a target is in P or starts by pq. In the following, we will associate to the i-th child of a pitch P the two corresponding paths p_i and q_i (which are uniquely determined).

In order to visit the search tree $\mathcal{T}$ in a depth-first fashion without storing its nodes in a stack (and thus saving space), we also define a linear-time computable function `next`, that allows us to jump from a child Q_i of a pitch P, corresponding to the paths (p_i, q_i), to the next child Q_{i+1}: in particular, $\texttt{next}(P, p_i, q_i) =$

(p_{i+1}, q_{i+1}). If Q_i is the last child of P, then the function returns the empty pair. Moreover, if the function `next` is invoked with arguments P and the two paths p and q that make P the child of its father, then it returns the pairs (p_1, q_1) corresponding to the first child of P (if it exists). By using the `father` and the `next` functions, we can then implement a depth-first search of the pitch tree $\mathcal{T}$ as shown in Fig. 2, where the dotted arcs denote a child relation. The rest of this section is devoted to the definition of the two functions `father` and `next` and to the proof of their time and space complexity.

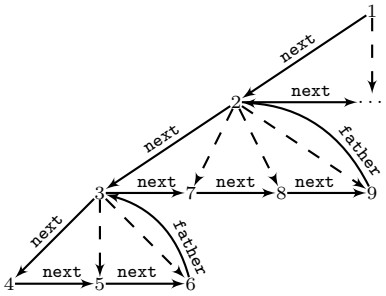

Fig. 2. The depth-first visit of the pitch tree

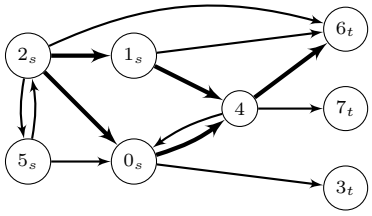

Fig. 3. An example of a pitch (in bold): the subscript indicates if a vertex is a source or a target

2.1 The `father` Function

The idea behind the definition of the child relationship (and thus of the pitch tree $\mathcal{T}$) is to build all pitches (starting from the empty one, which is the root of $\mathcal{T}$) by repeatedly adding paths from a source in S to a target in T (in short, *st*-paths). Since we want $\mathcal{T}$ to be a tree, we have to specify in which order *st*-paths are added, so that each pitch has a unique father. To this aim, we fix an arbitrary ordering of V, we lexicographically sort *st*-paths (which are ordered sequences of vertices), and impose that each new (explicitly or implicitly) added *st*-path is bigger than all *st*-paths included in the current pitch. In order to make this approach work, we need to overcome some problems, as shown in the following example.

Example 1. Let us consider the bold pitch shown in Fig. 3, which is formed by the *st*-paths $(0, 4, 6)$, $(1, 4, 6)$, $(2, 0, 4, 6)$, and $(2, 1, 4, 6)$ (listed in lexicographic order). If we allow any *st*-path bigger than $(2, 1, 4, 6)$ to be added to the pitch, then some problems might arise.

1. If we add $(2, 1, 4, 7)$, we are implicitly adding $(0, 4, 7)$, which is smaller than $(2, 1, 4, 6)$, contradicting the uniqueness of the father relationship, since the same pitch will also be reached through the explicit addition of $(0, 4, 7)$.
2. Sometimes it is necessary to implicitly add an *st*-path "much bigger" than the ones already present in the current pitch: adding $(2, 5, 0, 4, 6)$, we also add $(5, 0, 4, 6)$, thus eliminating the possibility of subsequently adding $(2, 6)$. Conversely, if we add first $(2, 6)$, then $(2, 5, 0, 4, 6)$ cannot be added. Therefore a pitch containing both $(2, 6)$ and $(2, 5, 0, 4, 6)$ will be missed.

In order to deal with the second problem, we need to specify that the *st*-paths that can be added to a pitch satisfy the following definition.

Definition 1. *A path $(p_0, \dots, p_k)$ of a pitch P is a component of P if it satisfies the following conditions: (1) $p_0 \in S$ is a source and $p_k \in T$ is a target, and (2) p_0 is not reachable in P from a smaller source in P (i.e., there is no $s \in P \cap S$ such that p_0 is reachable in P from s).*

Note that every arc in a pitch belongs to a component. Hence, a pitch can be specified by listing the set of its components: in particular, our pitch enumeration algorithm will proceed in lexicographic order with respect this time to the set of components.

Definition 2. *Given a pitch P, a child of P is a pitch $Q = P \cup c$ where c satisfies the following conditions: (1) c is the smallest component in Q which is not in P; and (2) c is bigger than any component in P.*

Example 2. Let us consider again the bold pitch P shown in Fig. 3. According to the above definition, the subtree of $\mathcal{T}$ rooted at P starts as shown in Fig. 4, where the labels of the edges denote the added component c. Observe that adding a component can cause the implicit addition of other components: for example, adding $(5, 2, 0, 4, 6)$ results also in adding $(5, 2, 1, 4, 6)$, which is however greater than $(5, 2, 0, 4, 6)$. Note also that $(2, 1, 4, 7)$ cannot be added to P, since it would not be the smallest new component not in P.

The following lemma shows that the child relationship defined above sorts all pitches in a tree with root the empty pitch.

Lemma 1. *Every pitch Q, apart from the empty one, has a unique father P.*

Proof. We start with the uniqueness: let us suppose $Q = P \cup c$ where P and c satisfy the conditions of Def. 2. We now show how c can be split into three paths, $c = pqp'$, where p is contained in P, q is "outside" of P (in the sense that it has no arc and no internal vertex in P), and p' is the remaining part of c (we denote by v and w respectively the start and the end of q). By the first condition of Def. 2, p' must be the smallest path in P from w to a target vertex in $T \cap P$: hence, $P = Q - q$ (see Fig. 1). This means that each internal vertex

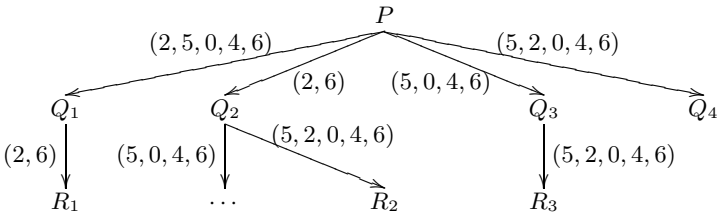

Fig. 4. A fragment of a pitch tree

in q has in-degree and out-degree equal to 1 in Q. Moreover, p is the only path from a source in $S \cap P$ to v: indeed, a smaller one would contradict the first condition of Def. 2, while a bigger one would contradict the second condition. This means that pq is an initial segment of the smallest component of Q which is not in P and that no vertex x in pq before w verifies any of the following conditions: (1) x has at least two incoming edges in Q; (2) x is a source smaller than the first vertex of p; and (3) x has no outgoing edge in Q. Since $Q - q$ is a pitch, w has to satisfy one of the conditions above and this characterizes w. We now need to characterize v. Since no internal vertex in q is in P, v must be the first vertex before w verifying one of the following conditions: (a) v has at least two outgoing edges in Q; (b) v is a target; and (c) v has no incoming edge in Q. Since both v and w are uniquely determined, we have that also q is uniquely determined: hence, the uniqueness is proved. In order to prove the existence, it is enough to show that $P \cup c$ is a child of P, where $c = pqp'$ and p, q and p' are the paths determined by the previous conditions. We have that c is bigger than every component in P because pq is a prefix of the last component of Q and q is outside P. Moreover, c satisfies the first condition of Def. 2 because each component of Q not in P contains an arc in q. This means that it contains the whole pq because of the conditions on v and w. □

From the proof of the previous lemma, the next result immediately follows.

Corollary 1. *It is possible to find in linear time the father of a pitch.*

Instead, the next consequence of the above lemma will yield a more "algorithmic" definition of the child relationship (see Fig. 1).

Corollary 2. *Let P and Q be two pitches. Q is a child of P if and only if $Q = P \cup pq$ where:*

- *pq is bigger than the last path in P;*
- *$p \subseteq P$;*
- *q is "outside" $P \cup T$, i.e. q is disjoint from P and $q_1, \ldots, q_{|q|-1}$ are not in $P \cup T$;*
- *no vertex in p satisfies Conditions 1-3 in the proof of Lemma 1.*

Moreover, p and q are uniquely determined.

Proof. If Q is a child of P, the proof of Lemma 1 implies all the conditions required and that p and q are uniquely determined. For the other direction, let p' be the first path in P from the end of q to a target. The path pqp' satisfies all conditions required in the child definition. □

2.2 The `next` Function

As we already said before, the `next` function should allow us to compute the first child Q_1 of P (if it exists) and the next child Q_{k+1} from the child Q_k (if it exists). In order to define this function well, we first prove the following result.

Lemma 2. *For any pitch P, the function*

$$\begin{array}{ccc} \Phi_P : \{\textit{children of } P\} & \to & \{\textit{paths in } G \textit{ satisfying Corollary 2}\} \\ P \cup q & \mapsto & (p, q) \end{array} \tag{1}$$

is an order-preserving bijection (pitches are sorted lexicographically as sets of components).

Proof. The function is well defined because of Corollary 2 and it is a bijection because the inverse function is $\Psi_P(p, q) := P \cup q$. It preserves the order because pq is a prefix of the first component Q not in P and a path satisfying Corollary 2 is never a prefix of another component (by the 3rd condition in Corollary 2). □

The function `next` is then defined as follows. Given a graph $G = (V, E, S, T)$, a pitch P, and a path r of G starting from a source which is a prefix of the last path in P or is bigger than every component of P, the function `next` returns the smallest path pq such that P and pq satisfy all conditions in Corollary 2, and pq is strictly bigger than r.

Theorem 1. *The* `next` *function is computable in time* $\mathcal{O}(|G|)$.

2.3 Complexity Analysis

The pitch enumeration algorithm is based on a depth-first search of the pitch tree $\mathcal{T}$, which uses the two functions `father` and `next`. Because of Corollary 1 and Theorem 1, every node can be visited in linear time. By the well-known *alternative output* technique [5, Theorem 1], it is possible to output a solution every time two nodes are visited, to obtain linear delay. To do so, all solutions with even depth in $\mathcal{T}$ must be output as soon as they are found, while solutions with odd depth must be output before computing their father. Since the depth changes by 1 every time a node is visited, the previous condition is accomplished. It is also easy to show that only a linear amount of space to store G, P and $r = (p, q)$ is required.

3 Enumerating Stories

In order to enumerate all the stories contained in a graph $G = (V, E, S, T)$, the approach described in [1] was based on generating all permutations of the vertices, and on cleaning and completing the corresponding DAG in order to turn it into a story. By using the results in the previous section instead, all the stories can be enumerated by enumerating all the pitches and outputting only the maximal ones. However, even if, in many real cases, this approach already outperforms the method proposed in [1], the method itself fails in enumerating, within a reasonable amount of time, all the stories in the case of large graphs. Indeed, usually an exponential number of pitches that are not stories can be generated. In order to avoid the computation of many useless pitches, we will now show how very often it is possible to verify *a priori* whether a pitch can lead to a story, thus performing a *pruning* of the pitch tree $\mathcal{T}$. In order to explain the pruning process, we introduce the following definitions.

Definition 3. *A pitch is a successor of $P \cup r$ if it is a descendant of P bigger than $P \cup r$ (pitches are sorted lexicographically as sets of components).*

Definition 4. *Given a pitch P and a path r, a vertex $v \in V$ is (P, r)-open if it belongs to a successor of $P \cup r$. A vertex is (P, r)-closed if it is not (P, r)-open.*

For example, the vertex 3 of Fig. 3 is $(P, (2, 5, 0, 4, 6))$-closed.

Lemma 3. *Let G be strongly connected and let P be a non-empty pitch. If $(S \cup T) - P \neq \emptyset$, then P is not a story.*

Proof. Assume there exists $s \in S - P$ and let p be a shortest path from s to any vertex in P (this path exists since G is strongly connected). Then $P \cup p$ is a pitch strictly containing P: this proves that P is not a story. Analogously, we can prove that if there exists $t \in T - P$, then P is not a story. □

Corollary 3. *Given a pitch P and a path r, if a source or a target is (P, r)-closed, no successor of $P \cup r$ is a story.*

For example, in the case of the fragment of a pitch tree shown in Fig. 4, we have that the subtrees rooted at Q_2, Q_3, and Q_4 do not contain any story (since the vertex 3 of Fig. 3 is $(P, (2, 5, 0, 4, 6))$-closed). Actually, vertex 3 is closed with respect to the empty pitch (which is the root of $\mathcal{T}$), and the path $(0, 4, 6)$, hence P will not even be reached since the pruning will be effective on the very first branch from $\emptyset$ to P. We may now state the main theorem used to prune the tree of all pitches.

Theorem 2. *Given a pitch P and a path r such that there exists a story which is a successor of $P \cup r$, there is no path p that verifies the following conditions: (1) $P \cup p$ is a pitch; (2) the last vertex of p is not in r; and (3) p is "outside" any successor of $P \cup r$, that is, no arc of p is in a successor of $P \cup r$ and all internal vertices of p are (P, r)-closed.*

Proof. Let Q be a successor of $P \cup r$ which is a story (Q exists by hypothesis). By the third condition on p, it follows that p is outside Q. Moreover, $Q \cup p$ is not a pitch (since Q is maximal): hence, there must be a path q in Q from the last vertex of p to the first one. By the first condition on p, it follows that q is not in P (since $P \cup p$ is acyclic). Consider now a path in P from a source to the last vertex of p (this path exists because of the second condition on p) and link this path to q: this can be extended to a component of Q. By the second condition on p, this component is smaller than r: this is a contradiction because this component is not in P (since it contains q which is not in P). □

The above theorem gives us a powerful tool to prune the tree of all pitches. Indeed, given a pitch P and a path r, if we can find a path p satisfying the three conditions of the theorem, we can then conclude that there is no story which is a successor of $P \cup r$. However, in order to apply this pruning criterion, we should be able to compute the set of vertices which are (P, r)-closed (or, equivalently,

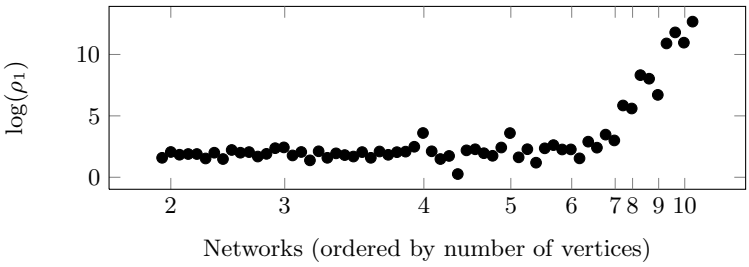

Fig. 5. Ratio between time consumed by Gobbolino and Touche to compute all stories in input graphs with 2 to 10 vertices (logarithmic scale)

the set of vertices which are (P, r)-open). So far, we have not been able to solve this latter problem (indeed, we conjecture it is NP-hard), but we can efficiently "approximate from above" the set of (P, r)-open vertices, that is, we can compute in linear time a superset of this set, which in practice is not too much bigger. Thanks to this result and to Theorem 2, we obtain an algorithm that decides if it is possible to prune the pitch tree in time $\mathcal{O}(|V||G|)$. The efficiency of this pruning process will be experimentally validated in the next section.

4 Experimental Results

In order to evaluate the efficiency of the new algorithm for the enumeration of stories, called Touche, we performed three experiments, two of them comparing with the previous algorithm proposed in [1], called Gobbolino, and the third one to evaluate the effect of the pruning approach (the entire dataset, the Java code, and the detailed experimental results are available starting from `amici.dsi.unifi.it/lasagne/`).

Enumerating All Stories
Our first experiment consisted in the enumeration of the whole set of stories using both Gobbolino and Touche (with the pruning approach implemented) and the comparison of their running time. Gobbolino is guaranteed to find all stories only if all permutation orderings of the vertices of the input graph are inspected, which limits its application to small input graphs. In [6], Gobbolino was applied in order to automatically recover the so-called *metabolic pathways* in a dataset consisting of 69 such pathways, among which 62 represented an input graph with no more than 10 vertices. For this subset, we obtained the results summarized in Figure 5. Let $t_{\mathtt{G}}(G)$ (resp., $t_{\mathtt{T}}(G)$) denote the time consumed by Gobbolino (resp., Touche) to compute all stories in the graph G, and let $\rho_1(G) = t_{\mathtt{G}}(G)/t_{\mathtt{T}}(G)$. In the figure we show the logarithm of ρ_1 for all the 62 graphs, ordered in increasing order with respect to their number of vertices. As it can be seen from the figure, Touche performs better than Gobbolino

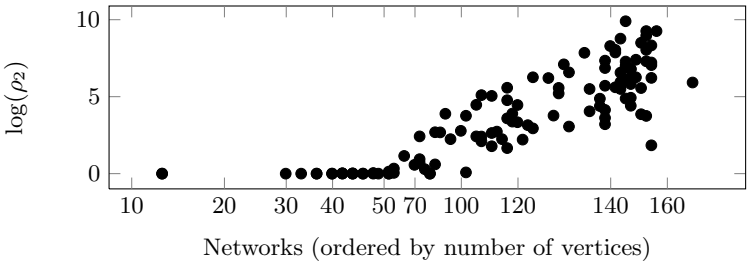

Fig. 6. Ratio between number of stories produced by GOBBOLINO and TOUCHE after 1 minute of computation (logarithmic scale)

for the whole dataset (even if the size of the instances is very small). Clearly, GOBBOLINO consumes more time as the size of the input increases, since it has to check all orderings of the vertices. For inputs with up to 7 vertices, both algorithms finish the enumeration process in less than 1 second. For the three inputs of size 8, GOBBOLINO consumes between 3.8 and 4.7 seconds, while TOUCHE never uses more than 0.05 seconds of computation. The result is even more impressive when we look at the inputs of size 9 and 10. GOBBOLINO takes around 1 minute for the three inputs of size 9 and more than 15 minutes for the input with 10 vertices, while TOUCHE finishes processing them in no more than 0.14 seconds. Indeed, the figure suggests that the ratio ρ_1 increases as an exponential with respect to the number of vertices, in the case of networks with at least 6 vertices (in the case of smaller networks, file management overhead has to be taken into account).

Sampling Stories

One approach used in [6] in order to apply GOBBOLINO for bigger inputs was to use random permutations of the orderings of the vertices to sample the space of pitches and, therefore, the space of solutions (*i.e.*, stories). Our second experiment consisted in comparing this randomized approach of GOBBOLINO to TOUCHE (with the pruning approach implemented), giving a fixed amount of time for both algorithms (1 minute, in our experiment) and checking how many stories each method produced. For this experiment, we selected 118 metabolic networks of various sizes. The dataset may be divided as follows: 8 networks (with size greater than or equal to 10) come from the same metabolic pathways considered in the first experiment; 4 networks are inputs for some experiments also performed in the context of [6] and for which the set of black vertices came from biological experiments; the remaining 106 were metabolic networks downloaded from the public database MetExplore ([8]) and with a random set of black vertices (5% of the vertices of the graph were considered to be black). For this dataset, we obtained the results summarized in Figure 6. Let $s_{\mathtt{G}}(G)$ (resp., $s_{\mathtt{T}}(G)$) denote the number of stories produced by GOBBOLINO (resp., TOUCHE) with input the graph G after 1 minute of computation, and let $\rho_2(G) = s_{\mathtt{T}}(G)/s_{\mathtt{G}}(G)$.

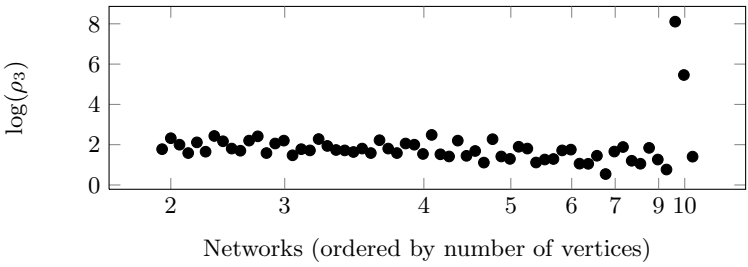

Fig. 7. Ratio between time consumed by TOUCHE to compute all stories without and with pruning

In the figure we show the logarithm of ρ_2 for all the 118 graphs, ordered in increasing order with respect to their number of vertices. The first outcome of this experiment is that TOUCHE always computed a number of stories bigger than or equal to the number of stories computed by GOBBOLINO. For 19 of them (mostly small size ones), the number is the same, but the time spent by TOUCHE is smaller than the limit of 1 minute, which indicates that the number of stories computed is in fact the total number of stories: this highlights another advantage of TOUCHE over the randomized version of GOBBOLINO, that continues exploring permutations of pitches even if it has already computed the whole set of stories. Moreover, note that TOUCHE produces the entire set of stories in the case of 10 other networks. In the case of bigger instances, the number of stories found by TOUCHE could be up to 950 times the number of stories found by GOBBOLINO. Indeed, the figure suggests that the ratio ρ_2 increases as an exponential with respect to the number of vertices. The extreme case is the `YERYP364` network for which GOBBOLINO found 4 stories while TOUCHE found 3815 stories: this result strongly suggests that the correspondence between permutation of the vertices and stories is highly biased and that there might be stories corresponding to very few permutations and hence unlikely to be produced by GOBBOLINO.

Evaluating the Pruning Methods

Our third experiment was designed in order to evaluate how effective is the pruning approach described in the previous section. By referring to the dataset used in the first experiment, for each network we collected the running time of TOUCHE with and without the pruning. The results are summarized in Figure 7. Let $t_{\mathtt{T},\mathtt{n}}(G)$ (resp., $t_{\mathtt{T},\mathtt{y}}(G)$) denote the time consumed by TOUCHE without (resp., with) pruning to compute all stories in the graph G, and let $\rho_3(G) = t_{\mathtt{T},\mathtt{n}}(G)/t_{\mathtt{T},\mathtt{y}}(G)$. In the figure we show the logarithm of ρ_3 for all the networks, ordered in increasing order with respect to their number of vertices. As it can be seen from the figure, TOUCHE with pruning always performs better than TOUCHE without pruning (even if the size of the instances is very small). The improvement seems to remain constant, even though in the case of two networks (that is, `PRPP-PWY` and `THREOCAT2-PWY`) it is quite impressive: the pruning improves the computational time by a factor of 275 in the first case and 43 in the second case.

Finally, we repeat this experiment in the case of two further networks analyzed in [6]: the first contains 10 vertices (8 black) and 222 stories and TOUCHE with pruning computed them about 5 times faster, the second contains 35 vertices (21 black) and TOUCHE with pruning computed its 3,934,160 stories in about three hours while TOUCHE without pruning did not finish after one day.

5 Conclusion

We presented a linear-time delay enumeration algorithm for pitches, that allowed us to enumerate all stories more efficiently than the previous known method [1]. The main question left open by our paper is to determine the complexity of the story enumeration problem.

Acknowledgements. The research leading to these results was funded by: the European Research Council under the European Community's Seventh Framework Programme (FP7 / 2007-2013) / ERC grant agreement n [247073]10; the French project ANR MIRI BLAN08-1335497; and the ANR funded LabEx ECOFECT.

References

1. Acuña, V., Birmelé, E., Cottret, L., Crescenzi, P., Jourdan, F., Lacroix, V., Marchetti-Spaccamela, A., Marino, A., Milreu, P.V., Sagot, M.F., Stougie, L.: Telling stories: Enumerating maximal directed acyclic graphs with a constrained set of sources and targets. Theor. Comput. Sci. 457, 1–9 (2012)
2. Schwikowski, B., Speckenmeyer, E.: On enumerating all minimal solutions of feedback problems. Discrete Applied Mathematics 117(1-3), 253–265 (2002)
3. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: On Generating All Maximal Independent Sets. Inf. Process. Lett. 27(3), 119–123 (1988)
4. Avis, D., Fukuda, K.: Reverse Search for Enumeration. Discrete Applied Mathematics 65, 21–46 (1993)
5. Uno, T.: Two general methods to reduce delay and change of enumeration algorithms. NII Technical Report (2003)
6. Milreu, P.V.: Enumerating Functional Substructures of Genome-Scale Metabolic Networks: Stories, Precursors and Organisations. PhD thesis, Université Claude Bernard, Lyon 1, France (2012)
7. Milreu, P.V., Acuña, V., Birmelé, E., Borassi, M., Cottret, L., Junot, C., Klein, C., Marchetti-Spaccamela, A., Marino, A., Stougie, L., Jourdan, F., Lacroix, V., Crescenzi, P., Sagot, M.-F.: Metabolic stories: exploring all possible scenarios for metabolomics data analysi (in preparation, 2013)
8. Cottret, L., Wildridge, D., Vinson, F., Barrett, M.P., Charles, H., Sagot, M.F., Jourdan, F.: Metexplore: a web server to link metabolomic experiments and genome-scale metabolic networks. Nucleic Acids Research 38(Web-Server-Issue), 132–137 (2010)
9. Johnson, C.H., Gonzalez, F.J.: Challenges and opportunities of metabolomics. Journal of Cellular Physiology 227(8), 2975–2981 (2012)