algorithms

MDPI

Article

MultiAspect Graphs: Algebraic Representation and Algorithms

Klaus Wehmuth [1,*], Éric Fleury [2] and Artur Ziviani [1,*]

[1] National Laboratory for Scientific Computing (LNCC), Av. Getúlio Vargas 333, 25651-075 Petrópolis, Brazil
[2] Institut national de recherche en informatique et en automatique (INRIA), Centre national de la recherche scientifique (CNRS), Université de Lyon, École normale supérieure de Lyon, UCB Lyon 1, LIP UMR 5668, F-69342, 46, Alleé d'Italie, 69364 Lyon, France; eric.fleury@ens-lyon.fr
* Correspondence: klaus@lncc.br (K.W.); ziviani@lncc.br (A.Z.); Tel.: +55-24-2233-6199 (A.Z.)

Academic Editor: Natarajan Meghanathan
Received: 25 September 2016; Accepted: 19 December 2016; Published: 25 December 2016

Abstract: We present the algebraic representation and basic algorithms for MultiAspect Graphs (MAGs). A MAG is a structure capable of representing multilayer and time-varying networks, as well as higher-order networks, while also having the property of being isomorphic to a directed graph. In particular, we show that, as a consequence of the properties associated with the MAG structure, a MAG can be represented in matrix form. Moreover, we also show that any possible MAG function (algorithm) can be obtained from this matrix-based representation. This is an important theoretical result since it paves the way for adapting well-known graph algorithms for application in MAGs. We present a set of basic MAG algorithms, constructed from well-known graph algorithms, such as degree computing, Breadth First Search (BFS), and Depth First Search (DFS). These algorithms adapted to the MAG context can be used as primitives for building other more sophisticated MAG algorithms. Therefore, such examples can be seen as guidelines on how to properly derive MAG algorithms from basic algorithms on directed graphs. We also make available Python implementations of all the algorithms presented in this paper.

Keywords: complex network; multilayer network; time-varying network; high order network; graph algorithms

1. Introduction

Graph theory finds many applications in the representation and analysis of complex networked systems [1–3]. In most cases, the utility of the graph abstraction comes from its inherent ability to represent binary transitive relations (i.e., transitive relations between two objects), which due to the transitivity property gives raise to key concepts, such as walks, paths, and connectivity. This graph conceptual framework allowed the emergence of basic algorithms, such as Breadth First Search (BFS) and Depth First Search (DFS) [4,5]. These basic graph algorithms, in their turn, allow the development of more sophisticated algorithms for the analysis of specific properties of complex networks, such as algorithms for computing some shortest-path-based network centralities or algorithms for analyzing network robustness [6–9]. Additionally, these basic graph algorithms also allow the analysis of dynamic processes in complex networks, such as network generative processes or information diffusion [10–13]. Moreover, several generalizations of the basic graph concept have been proposed for modelling complex systems that can be represented by layers of distinct networks [14,15] and also complex systems in which the network itself evolves with time [16,17].

In our previous work [18], we formalize the MultiAspect Graph (MAG) structure, while also stating and proving its main properties. The adopted adjacency concept in MAGs is similar to the one found in simple directed graphs, where the adjacency is expressed between two vertices, leading to

a structure in which an edge represents a binary relation between two composite objects. Moreover, in [18], we show that MAGs are closely related to simple directed graphs, as we prove that each MAG has a simple directed graph, which is isomorphic to it. This isomorphism relation between MAGs and directed graphs is a consequence of the fact that both MAGs and directed graphs share a similar adjacency relation.

MAGs find application in the representation and analysis of dynamic complex networks, such as multilayer or time-varying networks; or even networks that are both multilayer and time-varying as well as higher-order networks [19,20]. Examples of such networks include face-to-face in-person contact networks [21], mobile phone networks [22,23], gene regulatory networks [24], urban transportation networks [25], brain networks [26,27], social networks [28], among many others. In particular, we have previously applied the MAG abstraction from [18] to different purposes, such as modeling time-varying graphs [29], studying time centrality in dynamic complex networks [30], and investigating social events based on mobile phone networks [31]. To illustrate the MAG concept in more details in this paper, we present in Section 3 an example of modeling a simple illustrative multimodal urban transportation network.

In this paper, we build upon the basic MAG properties presented in [18] and show that MAGs can be represented by matrices in a form similar to those used for simple directed graphs (i.e., those with no multiple edges). Moreover, we here show that any algorithm (function) on a MAG can be obtained from its matrix representation. We then present the most common matrix representations that can be applied to MAGs, although we do not detail all the properties of these matrices, since they are well established in the literature [5,32]. Further, we introduce in detail the construction of MAG algorithms for computing degree, BFS, and DFS to exemplify how MAG algorithms can be derived from traditional graph algorithms, thus providing an illustrative guideline for developing other more sophisticated MAG algorithms in a similar way. Given that a MAG is isomorphic to a directed graph as shown in [18], the algebraic representation and algorithms for MAGs discussed in this paper closely resemble those for directed graphs. This isomorphism between MAGs and directed graphs constitutes an important theoretical result from [18] that paves the way for adapting well-known graph algorithms for application in MAGs, thus easing the effort to develop the analysis and application of MAGs for modelling complex networked systems, as we discuss in this paper. Therefore, the particular contribution of this paper is to discuss and to show how to build algebraic representations and algorithms for the MAG abstraction (topics that are not covered in the original MAG paper), bringing theoretical results related to these contributions. As a further contribution, we also make available Python implementations of all the algorithms presented in this paper at the following URL: http://github.com/wehmuthklaus/MAG_Algorithms.

This paper is organized as follows. Section 2 briefly presents the basic MAG definitions and properties derived from [18] in order to allow enough background of the current paper. Section 2 also presents illustrative examples of MAGs and its adjacency notion. Section 3 shows the representation of MAGs by means of algebraic structures, such as matrices. Emphasis is given to matrix representations, which are derived from the isomorphism relation between MAGs and simple directed graphs. In particular, we also introduce in Section 3.1 the companion tuple, which is a complement to the MAG matrix representations. In Section 4, we present basic MAG algorithms which are derived from well-known simple graph algorithms. Further, in Section 4.2, we show that any algorithm (function) that can be defined for a MAG can be also obtained from its adjacency matrix and companion tuple, establishing the theoretical basis for deriving MAG algorithms from well-known simple graph algorithms. Finally, Section 5 presents our final remarks and perspectives for future work.

2. MultiAspect Graph (MAG)

In this section, we present a formal definition of a MAG, as well as some key properties, which are formally stated and proved in [18].

2.1. MAG Definition

We define a MAG as $H = (A, E)$, where E is a set of edges and A is a finite list of *aspects*. Each aspect $\varphi \in A$ is a finite set, and the number of aspects $p = |A|$ is called the order of H. Each edge $e \in E$ is a tuple with $2 \times p$ elements. All edges are constructed so that they are of the form $(a_1, \ldots, a_p, b_1, \ldots, b_p)$, where a_1, b_1 are elements of the first aspect of H, a_2, b_2 are elements of the second aspect of H, and so on, until a_p, b_p which are elements of the p-th aspect of H. Note that the ordered tuple that represents each MAG edge is constructed so that their elements are divided into two distinct groups, each having exactly one element of each aspect, in the same sequence as the aspects are defined on the list A.

As a matter of notation, we say that $A(H)$ is the aspect list of H and $E(H)$ is the edge set of H. Further, $A(H)[n]$ is the n-th aspect in $A(H)$, $|A(H)[n]| = \tau_n$ is the number of elements in $A(H)[n]$, and $p = |A(H)|$ is the order of H.

In addition to the former definition, we define the following two sets constructed from the cartesian products of aspects of an order p MAG:

$$\mathbb{V}(H) = \bigtimes_{n=1}^{p} A(H)[n], \tag{1}$$

the cartesian product of all the aspects of the MAG H, and

$$\mathbb{E}(H) = \bigtimes_{n=1}^{2p} A(H)[(n-1)(mod\ p) + 1], \tag{2}$$

which is the set of all possible edges in the MAG H, so that $E(H) \subseteq \mathbb{E}(H)$.

We call $\mathbf{u} \in \mathbb{V}(H)$ a *composite vertex* of MAG H. As a matter of notation, a composite vertex is always represented as a bold lowercase letter, as in $\mathbf{u}$, for instance. From the properties stated for the MAG edge in our definition, it follows that an MAG edge is closely related to an ordered pair of composite vertices. For any given MAG H, every MAG edge $e \in E(H)$ has the form $(a_1, \ldots, a_p, b_1, \ldots, b_p)$, so that $(a_1, \ldots, a_p) \in \mathbb{V}(H)$ and $(b_1, \ldots, b_p) \in \mathbb{V}(H)$ are composite vertices of this given MAG H. From this, we can define two functions

$$\begin{aligned} \pi_o : \mathbb{E}(H) &\to \mathbb{V}(H) \\ e = (a_1, a_2, \ldots, a_p, b_1, b_2, \ldots, b_p) &\mapsto (a_1, a_2, \ldots, a_p) = \mathbf{u}, \end{aligned} \tag{3}$$

and

$$\begin{aligned} \pi_d : \mathbb{E}(H) &\to \mathbb{V}(H) \\ e = (a_1, a_2, \ldots, a_p, b_1, b_2, \ldots, b_p) &\mapsto (b_1, b_2, \ldots, b_p) = \mathbf{v}. \end{aligned} \tag{4}$$

We call $\pi_o(e)$ the origin composite vertex of e and $\pi_d(e)$ the destination composite vertex of e. Moreover, we can define the function

$$\begin{aligned} \psi : \mathbb{E}(H) &\to \mathbb{V}(H) \times \mathbb{V}(H) \\ e &\mapsto (\pi_o(e), \pi_d(e)) = ((a_1, \ldots, a_p), (b_1, \ldots, b_p)) = (\mathbf{u}, \mathbf{v}), \end{aligned} \tag{5}$$

from which we can construct a directed graph $G_H = (V(H), \psi(E(H))$. In [18], we demonstrate that the directed graph $G_H = (V(H), \psi(E(H))$ is isomorphic to the MAG H from which it was originated. At this point, we can therefore define the function

$$\begin{aligned} g : (A(H), E(H)) &\to (\mathbb{V}(H), \mathbb{V}(H) \times \mathbb{V}(H)) \\ H &\mapsto (\mathbb{V}(H), \psi(E(H)), \end{aligned} \tag{6}$$

which maps every MAG H to its isomorphic directed graph $g(H)$. Further, we define the set of functions

$$\begin{aligned} \pi_i : \mathbb{V}(H) &\to A(H)[i] \\ (a_1, a_2, \ldots, a_p) &\mapsto a_i, \end{aligned} \tag{7}$$

which extracts the n-th element of a composite vertex tuple.

2.2. MAG Sub-Determination

The sub-determination is a generalization of the aggregation concept applied to multilayer or time-varying graphs, in which all layers can be aggregated, resulting in a traditional graph. Since a MAG can have more than two aspects, the sub-determination can be done in more ways than the aggregation.

A given MAG H of order p, can be sub-determined in $2^p - 2$ ways. For each of these $2^p - 2$ ways, we have a list $A_C(H) \subset A(H)$ of the aspects used to determine an equivalence class. Note that in a MAG of order $p = 1$ (i.e., a traditional graph), a vertex can not be sub-determined, since $2^p - 2 = 0$.

2.2.1. Sub-Determined Composite Vertices

Let ζ, with $1 \leqslant \zeta \leqslant 2^p - 2$, be an index for one of the possible ways to construct a proper nonempty sublist of aspects. From this, we can define a canonical representation of the sub-determination directly defined by ζ. For any given ζ, we consider the p-bit binary expansion of ζ that is used as an indicator showing which aspects of the original MAG are present on the sub-determination. More specifically, the least significant bit indicates the presence or absence of the first aspect and the most significant bit indicates the presence or absence of the last aspect. By this convention, in a MAG with $p = 3$ aspects, we have that $\zeta = 001_2$ corresponds to the sub-determination where only the first aspect is present, $\zeta = 010_2$ corresponds to the sub-determination where only the second aspect is present, $\zeta = 101_2$ corresponds to the sub-determination where both the first and the third aspects are present, and so on. By using this convention, we can directly associate a given ζ to its corresponding aspect sublist.

Therefore, for each ζ, we have a unique sublist $A_\zeta(H)$ of aspects, such that $p_\zeta = |A_\zeta(H)|$ is the order of the sub-determination ζ. We now define the set

$$\mathbb{V}_\zeta(H) = \bigtimes_{n=1}^{p_\zeta} A_\zeta(H)[n], \tag{8}$$

where $\mathbb{V}_\zeta(H)$ is the cartesian product of all the aspects in the sublist $A_\zeta(H)$ of aspects, according to the index ζ. We call $\mathbf{u}_\zeta \in \mathbb{V}_\zeta(H)$ a sub-determined vertex, according to the sub-determination ζ.

We can now define the function

$$\begin{aligned} S_\zeta : \mathbb{V}(H) &\to \mathbb{V}_\zeta(H) \\ (a_1, a_2, \ldots, a_p) &\mapsto (a_{\zeta_1}, a_{\zeta_2}, \ldots, a_{\zeta_m}), \end{aligned} \tag{9}$$

where $m = p_\zeta$. S_ζ maps a composite vertex $\mathbf{u} \in \mathbb{V}(H)$ to the corresponding sub-determined composite vertex $\mathbf{u}_\zeta \in \mathbb{V}_\zeta(H)$, according to the sub-determination ζ. As $(a_{\zeta_1}, a_{\zeta_2}, \ldots, a_{\zeta_m}) \in \mathbb{V}_\zeta(H)$, it follows that $a_{\zeta_1} \in A_\zeta(H)[1], \ldots, a_{\zeta_m} \in A_\zeta(H)[m]$. From the definition, it can be seen that the function S_ζ is not injective. Hence, the function S_ζ for a given sub-determination can be used to define a equivalence relation $\equiv_\zeta$ in $\mathbb{V}(H)$, where for any given composite vertices $\mathbf{u}, \mathbf{v} \in \mathbb{V}(H)$, we have that $\mathbf{u} \equiv_\zeta \mathbf{v}$ if and only if $S_\zeta(\mathbf{u}) = S_\zeta(\mathbf{v})$.

2.2.2. Sub-Determined Edges

From the sub-determination ζ of order p_ζ, we can also construct the set

$$\mathbb{E}_\zeta(H) = \bigtimes_{n=1}^{2\times p_\zeta} A_\zeta(H)[(n-1)(mod\ p_\zeta)+1], \tag{10}$$

where $p_\zeta = |A_\zeta(H)|$ is the order of the sub-determination ζ, and $\mathbb{E}_\zeta(H)$ is the set of all possible sub-determined edges according to ζ. We then define the function

$$\begin{aligned} E_\zeta : \mathbb{E}(H) &\to \mathbb{E}_\zeta(H) \\ (a_1, a_2, \ldots, a_p, b_1, b_2, \ldots, b_p) &\mapsto (a_{\zeta_1}, a_{\zeta_2}, \ldots, a_{\zeta_m}, b_{\zeta_1}, b_{\zeta_2}, \ldots, b_{\zeta_m}), \end{aligned} \tag{11}$$

where $m = p_\zeta$ and $a_{\zeta_1}, b_{\zeta_1} \in A_\zeta(H)[1], a_{\zeta_2}, 2_{\zeta_2} \in A_\zeta(H)[2], \ldots, a_{\zeta_m}, b_{\zeta_m} \in A_\zeta(H)[m]$. This function takes an edge to its sub-determined form according to ζ in a similar way as defined above for composite vertices. In general, the function E_ζ is not injective. Consider two distinct edges $e_1, e_2 \in E(H)$, such that e_1 and e_2 differ only in aspects which are not in $A_\zeta(H)$. Since $E_\zeta(\cdot)$ only contains values for aspects present in $A_\zeta(H)$, it follows that $E_\zeta(e_1) = E_\zeta(e_2)$, and therefore E_ζ is not injective. Further, consider an edge $e \in E(H)$ and its sub-determined edge $e_\zeta = E_\zeta(e)$, such that $\pi_o(e_\zeta) = \pi_d(e_\zeta)$, i.e., e_ζ is a self-loop. Since self-loops are not allowed to be present on a MAG, it follows that $e_\zeta \notin E_\zeta(E(H))$. As consequence, we have that $|E_\zeta(E(H))| \leqslant |E(H)|$.

2.2.3. Sub-Determined MAGs

For a given sub-determination ζ we have the sublist $A_\zeta(H)$ of considered aspects and also the sub-determined edges obtained from ζ. Based on them, we can now obtain a sub-determined MAG. For a given sub-determination ζ we define the function

$$\begin{aligned} M_\zeta : (A(H), E(H)) &\to (A_\zeta(H), \mathbb{E}_\zeta(H)) \\ H &\mapsto (A_\zeta(H), E_\zeta(E(H))). \end{aligned} \tag{12}$$

Since $A_\zeta(H)$ is the sublist of aspects of H prescribed by ζ and $E_\zeta(E(H))$ is the set of all sub-determined edges according to the sub-determination ζ, it follows that $(A_\zeta(H), E_\zeta(E(H)))$ is a MAG obtained from H according to the sub-determination ζ. As $|A_\zeta(H)| < |A(H)|$, it follows that the order of $M_\zeta(H)$ is lower than the order of H. Further, since self-loops may be created by edge sub-determination and discarded, and also since E_ζ is not injective, it follows that $|E_\zeta(E(H))| \leqslant |E(H)|$.

2.3. MAG Adjacency

Two composite vertices are considered adjacent if they share the same MAG edge, i.e., given two composite vertices $\mathbf{u}, \mathbf{v} \in \mathbb{V}(H)$ are adjacent if and only if there is a MAG edge $e \in E(H)$ such that $\mathbf{u}, \mathbf{v} \in \{\pi_o(e), \pi_d(e)\}$. Similarly, two MAG edges are considered adjacent if and only if they share a same composite vertex, i.e., two given edges $e_1, e_2 \in E(H)$ are adjacent if and only if there is a composite vertex $\mathbf{u} \in \mathbb{V}(H)$ such that $\mathbf{u} \in \{\pi_o(e_1), \pi_d(e_1)\}$ and $\mathbf{u} \in \{\pi_o(e_2), \pi_d(e_2)\}$.

Figure 1 shows an illustrative example of three MAG edges. The figure depicts a four aspects MAG, where each set of colored circles represents one aspect, and each edge has two elements of each aspect.

The isolated edge $(A1, A2, A3, A4, C1, C2, C3, C4)$ on the leftmost side of Figure 1 exemplifies the composite vertex adjacency concept. In this case the composite vertices $(A1, A2, A3, A4)$ and $(C1, C2, C3, C4)$ are adjacent. The two edges $(E1, E2, E3, E4, H1, H2, H3, H4)$ and $(H1, H2, H3, H4, K1, K2, K3, K4)$ exemplify a case of edge adjacency. Since the composite vertex $(H1, H2, H3, H4)$ is shared by both edges, they are adjacent.

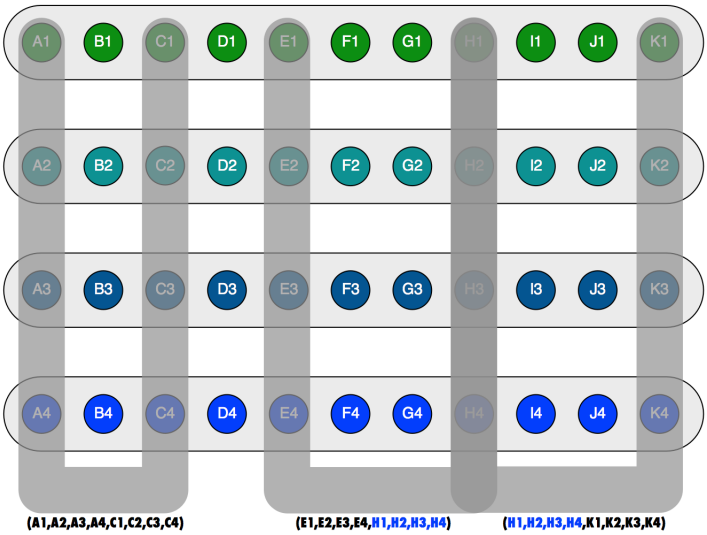


Figure 1. Illustrative example of three MultiAspect Graphs (MAG) edges.

Although the structure of a MAG edge is similar to an even uniform hypergraph edge, the adjacency definition used on MAGs is not the usual one adopted on hypergraphs. The adjacency concept found on a MAG is close to the one associated with traditional directed graphs, where a MAG edge can be seen as a relation between two composite vertices, which are composite objects constructed from aspect elements. Therefore, a MAG edge expresses a relationship between two (composite) objects in the same way as a directed graph edge. This concept leads to the isomorphism between MAGs and directed graphs, as well as the close relation between walks, trails, and paths on MAGs and directed graphs.

2.4. MAG Isomorphism

Before formally defining the MAG isomorphism, it is necessary to define the concept an aspect list bijection. Given two MAGs H and K, both with p aspects, an aspect list bijection $F : A(H) \rightarrow A(K)$ is defined as a set of p bijective functions, $f_1, f_2, \ldots, f_p$, such that each aspect of the MAG H is the domain of exactly one of these functions and each aspect of MAG K is the codomain of exactly one of these functions. It follows from this definition that given a composite vertex $\mathbf{u} \in \mathbb{V}(H)$, the aspect list bijection F takes $\mathbf{u}$ to a composite vertex $F(\mathbf{u}) \in \mathbb{V}(K)$.

Two MAGs of order p, H and K, are isomorphic if there is an aspect list bijection $F : A(H) \rightarrow A(K)$ such that an edge $e \in E(H)$ if and only if the edge $(F(\pi_o(e)), F(\pi_d(e))) \in E(K)$.

2.5. MAG Walks, Trails, and Paths

There is a close relation between walks, trails, and paths on a MAG and their counterparts in the isomorphic directed graph $g(H)$.

A walk on a MAG H is defined as an alternating sequence $W = [\mathbf{u}_1, e_1, \mathbf{u}_2, e_2, \mathbf{u}_3, \ldots, \mathbf{u}_{k-1}, e_{k-1}, \mathbf{u}_k]$ of composite vertices $\mathbf{u}_n \in \mathbb{V}(H)$ and edges $e_m \in E(H)$, such that $\mathbf{u}_n = \pi_o(e_n)$ and $\mathbf{u}_{n+1} = \pi_d(e_n)$ for $1 \leqslant n < k$. It follows from this definition that in a walk, consecutive composite vertices as well as consecutive MAG edges are adjacent.

We show in [18] that an alternating sequence W of composite vertices and edges in a MAG H is a walk on H if and only if there is a corresponding walk G_W in the composite vertices representation of H. This means that a walk on a MAG H has a isomorphic walk on the directed graph $g(H)$. Since trails and paths also are walks, we also show that the same isomorphism concept extends to them as well.

Figure 1 can also exemplify a MAG path. The two edges $(E1, E2, E3, E4, H1, H2, H3, H4)$ and $(H1, H2, H3, H4, K1, K2, K3, K4)$ can also be seen as part of the alternating sequence $P = (E1, E2, E3, E4), (E1, E2, E3, E4, H1, H2, H3, H4), (H1, H2, H3, H4), (H1, H2, H3, H4, K1, K2, K3, K4), (K1, K2, K3, K4)$, which characterizes a two-hops path from the composite vertex $(E1, E2, E3, E4)$ to the composite vertex $(K1, K2, K3, K4)$.

From the concept that walks, trails, and paths on a MAG have a isomorphism relation to their counterparts on the directed graph $g(H)$, it follows that analysis and algorithms based on walks, trails, and paths can be formulated on the directed graph $g(H)$. These properties will be extensively used in the current work.

3. Algebraic Representation

In this section, we discuss ways to represent MAGs [18] by means of algebraic structures. As a consequence to the isomorphism between MAGs and traditional directed graphs, it is straightforward to construct matrix-based representations of MAGs. This section addresses these representations, using the MAG depicted in Figure 2 as an illustrative example.

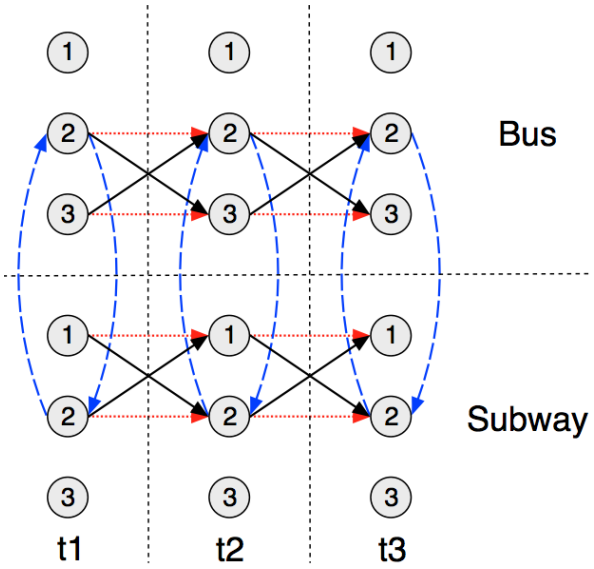


Figure 2. Illustrative MAG T of a hypothetical simple urban transit system.

Figure 2 shows an illustrative example of a three aspect MAG T. It can be seen as a representation of a time-varying multilayer network, showing a hypothetical simple urban transit system. More specifically, Figure 2 depicts the MAG T in its composite vertices representation, $g(T)$, which is the directed graph defined in Expression (6).

Aligned with this view, the aspects of MAG T can be interpreted in the following way: The first aspect represents three distinct locations, labeled 1, 2 and 3. Specifically, location 1 represents a subway station, location 2 a subway station with a bus stop, and location 3 a bus stop. The second aspect represents two distinct urban transit modes depicted as layers, namely Bus and Subway. Finally, the third aspect represents three time instants. The MAG edges can be seen with the following meaning: Location 1 has no edges on the bus transit mode, since it is a subway station. Similarly, location 3 has no edges on the subway mode, since it is a bus stop. The eight black edges represent bus and subway trips between locations. As a simplification all trips are assumed to have the same duration. The red (dotted) edges represent the possibility of staying at a bus stop or subway station and not taking a transit. The six blue (dashed) edges show that it is possible to change between bus and subway layers at all times at location 2. As a simplification, the connection between the bus and subway layers is assumed to take no time. We recognize that the decision of making these edges with 0 time length

generates cycles of length 0 in instances of location 2. In real network analysis, 0 length cycles (and also negative length cycles) can cause problems. However, we choose to let these cycles present in this toy example since they will cause no harm for the analysis conducted in this thesis, and also, they make the toy example more compact and readable. Further, we remark that if desired, these 0 length cycles could be broken by adding new composite vertices, or by making the subway/bus transition to have the same length as a subway/bus trip.

In this model, walks represent the ways the urban transit system can be used to travel from one location to another. For instance, starting at location 1 on the subway layer at time $t1$, it is possible to reach location 3 on the bus layer at time $t3$. It can be done by taking a subway trip to location 2 at time $t2$, switching from subway to bus layer at location 2, time $t2$ and finally taking a bus trip from location 2 bus layer arriving at location 3 on the bus layer at time $t3$.

The presence of unconnected occurrences of location 1 at bus layer and location 3 at subway layer can be viewed as artefacts of the MAG construction. We call these vertices trivial components of the MAG. This subject will be further addressed in this section.

3.1. Companion Tuple

Although we show that every MAG H is isomorphic to a directed graph designated $g(H)$, it is important to note that the set of vertices of this graph is $\mathbb{V}(H)$, as shown in Expression (6). Since the set $\mathbb{V}(H)$ is the cartesian product of all the aspects in the MAG H, it is possible to reconstruct the MAG's aspect list from $\mathbb{V}(H)$, which is a step necessary to obtain the MAG H from the directed graph $g(H)$. When the vertices of the directed graph G associated with a given MAG H are not the composite vertices themselves, it is necessary to provide a mechanism to link each vertex of the directed graph to its corresponding composite vertex on the MAG. This mechanism can be, for instance, a bijective function between $\mathbb{V}(H)$ and $V(G)$.

In the current work, we construct representations for $g(H)$, such as matrices, which do not directly carry the tuples that characterize the MAG's composite vertices. In this kind of representation, a vertex is associated with a row or column of a matrix. Therefore, additional information has to be provided to properly link each row (column) of a matrix to its corresponding composite vertex on the MAG represented by this matrix. This is done by a bijective function D, defined in Section 3.2, where D takes a composite vertex to a natural number, which is the row (column) number in the matrix.

The implementation of D presented in this work is based on the concept of a *companion tuple*, which complements the matrix representation of a given MAG. For a MAG H with p aspects, its companion tuple has the form $(|A(H)[1]|, |A(H)[2]|, \ldots, |A(H)[p]|)$, so that the number of elements on it equals the order of H and each element represents the number of elements of an aspect of H. As a matter of notation, we represent the companion tuple of a given MAG H as

$$\tau(H) = (|A(H)[1]|, |A(H)[2]|, \ldots, |A(H)[p]|), \tag{13}$$

where p is the order of H. When there is no ambiguity in relation to which MAG we are referring to, we may use the notation τ instead of $\tau(H)$. For instance, the companion tuple of the MAG T shown in Figure 2 is $\tau(T) = (3, 2, 3)$, since T has 3 aspects, of which the first has 3 elements, the second 2 elements, and the third 3 elements.

Algorithm 1 shows the building of the companion tuple for a given MAG H. Assuming that the size of the aspect list $A(H)$ and the size of each of the aspect sets contained in $A(H)$ are known from the computational representation of $A(H)$, the time complexity for building the companion tuple is $O(p)$, where p is the number of aspects on MAG H. If, however, these sizes are unknown, then the time complexity is $O(s)$, where $s = \sum_{i=1}^{p} |A(H)[i]|$, since each element of each aspect has to be counted. We remark that, in either case, the time complexity for building the companion tuple is less than $O(\mathbb{V}(H))$, which is the cardinality of the set of composite vertices of the MAG.

Algorithm 1: Construction of the companion tuple of a MAG.

input : $A(H)$
output: $\tau(H)$

CompTuple($A(H)$**)**
  $p \leftarrow |A(H)|$ `// number of aspects in the MAG`
  for $i \leftarrow 1$ ***to*** p **do**
    $T[i] \leftarrow |A(H)[i]|$ `// number of elements in` i`-th aspect`
  end
return T

For a given MAG H and a sub-determination ζ, we also define the sub-determined companion tuple $\tau_\zeta(H)$, which is obtained by multiplying each entry of the original companion tuple by the equivalent entry of the tuple representation of ζ, as shown in Algorithm 2. The sub-determined companion tuple has the same value as the original companion tuple for the aspects that have value 1 in ζ and 0 otherwise.

Algorithm 2: Construction of sub-determined companion tuple.

input : $\tau(H), \zeta$
output: $\tau_\zeta(H)$

SubCompTuple($\tau(H), \zeta$**)**
  $p \leftarrow |\tau(H)|$ `// number of aspects in the MAG`
  for $i \leftarrow 1$ ***to*** p **do**
    $T_\zeta[i] \leftarrow \tau(H)[i] * \zeta[i]$
  end
return T_ζ

3.2. Ordering of Composite Vertices and Aspects

In general, the sequence in which the composite vertices and aspects are organized on a MAG is not relevant. That is, changing the sequence in which the aspects or their elements are presented does not affect the result of any algorithm or analysis performed on a MAG, since the MAG obtained by such changes is isomorphic to the original one. The definition of the MAG isomorphism adopted in this work can be found in Section 2.4. However, to show the MAG's algebraic representation in a consistent way, it is necessary to link the MAG's composite vertices to rows and columns of matrices, which is achieved by the bijective function D, defined in this section at Equation (15). We now show the preliminary steps necessary for the definition of function D, as implemented in this work.

The aspect sequence is adopted as the same in which the aspects are placed on the MAG's companion tuple. For the ordering of composite vertices, we define the numerical representation of each composite vertex from its tuple. To obtain the composite vertex numerical representation, we first translate the composite vertex into a numerical tuple. This is done by applying a family of indices, one for each aspect on the composite vertex, where for every aspect i the corresponding index ranges from 0 to $\tau_i - 1$, where τ_i is the number of elements on the i-th aspect of the MAG. Since this is a simple index substitution, we do not use a distinct notation for the composite vertex on its numerical tuple form. We, however, reserve the notation $\mathbf{u}[i]$ to express the i-th element of the composite vertex on its numerical form.

To calculate the numerical representation of a composite vertex, we define the weight of each position on the composite vertex tuple of a MAG H with p aspects as

$$W(i,\tau) = \begin{cases} 1 & \text{if } i = 1, \\ \prod_{j=1}^{i-1} \tau_j & \text{otherwise,} \end{cases} \tag{14}$$

where i is the position in the tuple varying from 1 to p, τ is the MAG's companion tuple, and τ_j is the j-th element of the MAG's companion tuple. Note that $|\tau| = p$ meaning that the length of the companion tuple is the order of the MAG, i.e., the number of its aspects. Finally, we define the composite vertex numerical representation as

$$D(\mathbf{u},\tau) = 1 + \sum_{i=1}^{|\tau|} W(i,\tau) \times \mathbf{u}[i], \tag{15}$$

where $|\tau| = p$ is the MAG's order, and $\mathbf{v}[i]$ is the i-th component of the composite vertex. Figure 3 shows the MAG T with its composite vertices, and their numerical representations ranging from (1) to (18). To illustrate how the numerical representations are obtained, we show examples based on the MAG T.

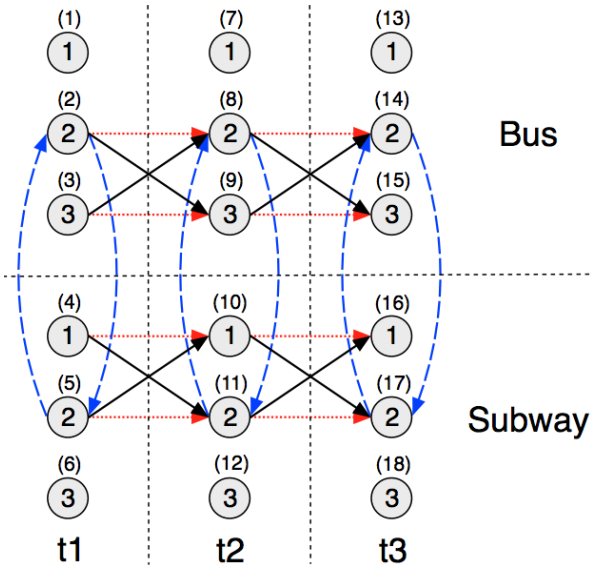


Figure 3. MAG T with composite vertices numerical representations.

For this representation, we adopt aspect indices such that for aspect 1 we have $Idx(1) = 0, Idx(2) = 1$ and $Idx(3) = 2$. For aspect 2, we have $Idx(Bus) = 0$ and $Idx(Subway) = 1$, while for aspect 3, $Idx(t_1) = 0, Idx(t_2) = 1$ and $Idx(t_3) = 2$. Since the companion tuple of MAG T is $\tau(T) = (3,2,3) = \tau$, the weights are $W(1,\tau) = 1$, $W(2,\tau) = \tau_1 = 3$ and $W(3,\tau) = \tau_1 \times \tau_2 = 6$. Therefore, the composite vertex $\mathbf{v} = (1, Bus, t_1)$ has numerical representation $D(\mathbf{v},\tau) = 1 + 1 \times 0 + 3 \times 0 + 6 \times 0 = 1$, while $D((2, Subway, t_2),\tau) = 1 + 1 \times 1 + 3 \times 1 + 6 \times 1 = 11$ and $D((2, Bus, t_3),\tau) = 1 + 1 \times 1 + 3 \times 0 + 6 \times 2 = 14$.

Algorithm 3 determines the numerical representation of a composite vertex v represented by its numerical tuple. The presented implementation extends the concepts presented in Equations (14) and (15), so that this algorithm can also be used to determine the numerical representation of sub-determined composite vertices. To determine the numerical representation of a sub-determined vertex, function D shown in Algorithm 3 receives the full composite vertex tuple (not sub-determined) and the sub-determined companion tuple. The **if** seen at line 7 of Algorithm 3 makes that the 0 entries found in a sub-determined companion tuple are discarded for the construction of the sub-determined

numerical representation of the composite vertex. The time complexity for this algorithm is $O(p)$, where p is the number of aspects on the MAG in question.

Algorithm 3: Determination of the numerical representation of a composite vertex.

```
   input  : v, τ(H)
   output : d
1                                  // v is the numerical tuple of the composite vertex
2  D(v, τ(H))
3      p ← |τ(H)|                                  // number of aspects in the MAG
4      d ← 0
5      w ← 1
6      for i ← 1 to p do
7          if τ(H)[i] ≠ 0 then
8              d ← d + (v[i] * w)
9              w ← w * τ(H)[i]
10         end
11     end
12 return d
```

Given the numerical representation of any composite vertex, it is possible to reconstruct its tuple. To do this, we calculate the numerical value of the index of each element on the tuple, as

$$N(d, i, \tau) = \lfloor ((d-1) \; mod \; W(i+1, \tau)) \; / \; W(i, \tau) \rfloor, \tag{16}$$

where d is the composite numerical representation, i is the position of the composite vertex tuple to be calculated, τ is the MAG's companion tuple, *mod* is the modulus (division remainder) operation and $\lfloor x \rfloor$ is the floor operator, which for any $x \in \mathbb{R}$ corresponds to the largest integer $i \in \mathbb{Z}$ such that $i \leqslant x$. Note that for calculating $N(d, p, \tau)$ for a MAG with p aspects, it is necessary to calculate $W(p+1, \tau)$. Considering the definition of W from Equation (14), it follows that $W(p+1, \tau) = \prod_{j=1}^{p} = |\mathbb{V}(H)|$, the number of composite vertices on the MAG.

For instance, taking the composite vertex with numerical representation 14 of the MAG T, we have that

$$\begin{aligned} N(14, 1, (3, 2, 3)) &= \lfloor ((14-1) \; mod \; 3)/1 \rfloor = \lfloor 1/1 \rfloor = 1 \\ N(14, 2, (3, 2, 3)) &= \lfloor ((14-1) \; mod \; 6)/3 \rfloor = \lfloor 1/3 \rfloor = 0 \\ N(14, 3, (3, 2, 3)) &= \lfloor ((14-1) \; mod \; 18)/6 \rfloor = \lfloor 13/6 \rfloor = 2. \end{aligned}$$

We can therefore define the inverse of function D as

$$D^{-1}(d, \tau) = (N(d, 1, \tau), N(d, 2, \tau), \ldots, N(d, |\tau|, \tau)), \tag{17}$$

which reconstructs the composite vertex tuple in its numerical form. From this, we can see that, for instance, $D^{-1}(14, (3, 2, 3)) = (1, 0, 2)$, which corresponds to the composite vertex $(2, Bus, t_3)$. Algorithm 4 shows the implementation of D^{-1}.

The relation between the composite vertex numerical representation and its tuple can also be seen as a consequence of the natural isomorphism between the MAG H and its composite vertices representation, $g(H)$. The role of this relation will become clear in Sections 3.4 and 3.5, where the matrix forms of the MAG are presented.

Algorithm 4: Determination of the composite vertex from its numerical representation.

```
   input  : d, τ(H)
   output : v
1                                  // v is the numerical tuple of the composite vertex
2  InvD(d, τ(H))
3      p ← |τ(H)|                                  // number of aspects in the MAG
4      w[1] ← 1
5      wl ← 1
6      for i ← 1 to p do
7          if τ(H)[i] ≠ 0 then
8              w[i + 1] ← w[i] ∗ τ(H)[i]
9              wl ← wl + 1
10         end
11     end
12     for i ← 1 to wl do
13         v[i] ← (d Mod w[i + 1])/w[i]
14     end
15 return v
```

3.3. Elimination of Trivial Components

In the MAG T shown in Figure 3 the composite vertices of numerical representation (1), (6), (7), (12), (13), and (18) are trivial components (i.e., unconnected composite vertices). They are created in consequence of the regularity needed on the MAG H to build the set $\mathbb{V}(H)$. This type of padding is not necessary in a directed graph and its algebraic representation. Therefore, it is possible to remove the trivial components from the composite vertices representation and its associated matrices. However, it is important to bear in mind that the graph resulting from this transformation may no longer be isomorphic to the MAG and neither are the matrices associated with it. The only case in which the isomorphism is preserved is when there are no trivial components on the MAG and nothing is removed. Nevertheless, this kind of transformation can be helpful for application, by reducing the number of composite vertices present on the graph and so simplifying its construction and manipulation.

The same sort of padding is discussed in [15], where authors suggest that this padding may cause problems in the computing of some metrics, such as mean degree or clustering coefficients, unless one accounts for the padding scheme in an appropriate way. In this subsection, we show that the padding with the trivial components may be eliminated, if desired. Anyway, if needed, it suffices to be cautious in computing the metrics of interest on MAGs by considering the existence of the padding scheme, as suggested by [15]. In particular, the MAG algorithms we discuss in Section 4 remain unaffected by this padding issue.

For a given MAG H, we define its main components graph $m(H)$ as the MAG's composite vertices representation with all its trivial components removed. Figure 4 shows the main components graph $m(T)$ for the MAG T. It is worth noting that numerical representations are not defined for $m(T)$.

This can be achieved algebraically for any MAG H with the help of a matrix $\mathbf{R}(H)$ constructed from the identity $\mathbf{I} \in \mathbb{R}^{n\times n}$, where $n = |\mathbb{V}(H)|$ is the number of composite vertices on the MAG. The matrix $\mathbf{R}(H)$ is obtained from this $n \times n$ identity by removing the columns which match the numerical representations of the trivial components of the MAG. Therefore, assuming that the MAG H has r trivial components, the matrix $\mathbf{R}(H) \in \mathbb{R}^{n\times n-r}$ has n rows and $n - r$ columns. In particular, in the cases where the MAG H has no trivial components, we have that $\mathbf{R}(H) = \mathbf{I} \in \mathbb{R}^{n\times n}$.

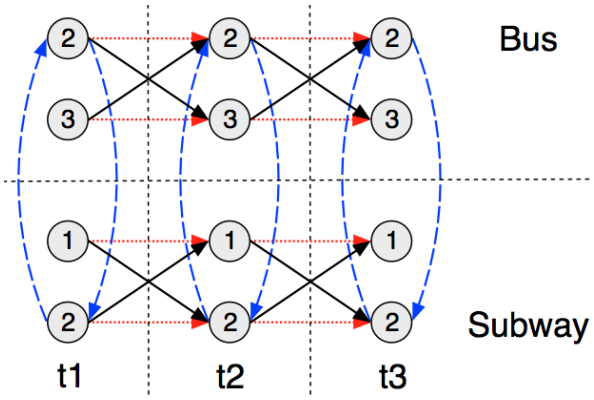


Figure 4. $m(T)$ of the example MAG T.

It is also worth noting that the matrix $\mathbf{I}_m(H) = \mathbf{R}(H)\,\mathbf{R}(H)^T \in \mathbb{R}^{n\times n}$ is a matrix akin to the identity $\mathbf{I} \in \mathbb{R}^{n\times n}$, but the diagonal entries corresponding to the trivial components (removed in $\mathbf{R}(H)$) have value 0. Therefore, multiplying a $n \times n$ matrix by $\mathbf{I}_m(H)$ to the left has the effect of turning all entries on the rows corresponding to the trivial components to 0 s. Similarly, multiplying by $\mathbf{I}_m(H)$ to the right has the effect of turning the entries of the columns corresponding to the trivial elements to 0 s.

As an example, we show the matrix $\mathbf{R}(T) \in \mathbb{R}^{18\times 12}$,

$$\mathbf{R}(T) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{18}$$

which is obtained from the 18×18 identity matrix by removing the columns $1, 6, 7, 12, 13$, and 18 that correspond to the trivial components of the MAG T.

The algebraic form of the elimination of the trivial components is presented in Equation (22) in Section 3.4, after the required definition of adjacency matrix.

3.4. Adjacency Matrix

As a direct consequence of the isomorphism between MAGs and traditional directed graphs, it is expected that a MAG can be represented in matrix form. In fact, such representations can be achieved directly by the composite vertices representation of MAGs, presented in Section 2.1. Since for any given MAG H its composite vertices representation is a traditional directed graph, it can be represented in matrix form.

One of such representations is the MAG's adjacency matrix. This matrix is obtained from the MAG's composite vertices representation, $g(H)$, and its companion tuple $\tau(H)$. In fact, the MAG's adjacency matrix is the adjacency matrix of the composite vertices representation, where the sequence of the rows and columns is given by the numerical representation of the composite vertices of $g(H)$.

Since the set $\mathbb{V}(H)$ of composite vertices of a given MAG H is obtained by the cartesian product of all aspects of the MAG (as shown in Expression (1)), it follows that the number of composite vertices on a given MAG H with p aspects is

$$n = |\mathbb{V}(H)| = \prod_{i=1}^{p} \tau_i, \tag{19}$$

where τ_i is the i-th element of the MAG's companion tuple, i.e., the number of elements on the MAG's i-th aspect.

The general form of any entry of the matrix $\mathbf{J}(H)$ is given by

$$j_{\mathbf{u},\mathbf{v}} = \begin{cases} 1 & \text{if } (\mathbf{u},\mathbf{v}) \in E(g(H)), \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

where $(\mathbf{u},\mathbf{v}) \in E(g(H))$ means that $(\mathbf{u},\mathbf{v})$ is an edge on the composite vertices representation $g(H)$ of the MAG H, so that $\mathbf{u},\mathbf{v} \in \mathbb{V}(H)$ are composite vertices of H. It follows from the definition of $g(H)$ and its natural isomorphism to H, that $(\mathbf{u},\mathbf{v}) \in E(g(H))$ if and only if there is an edge $e \in E(H)$ such that $\mathbf{u} = \pi_o(e)$ and $\mathbf{v} = \pi_d(e)$. It is important to note, however, that the notation $j_{\mathbf{u},\mathbf{v}}$ is in fact a shorthand for $j_{D(\mathbf{u},\tau),D(\mathbf{v},\tau)}$, where $D(\mathbf{u},\tau)$ is the row number and $D(\mathbf{v},\tau)$ the column number of the matrix entry. This ties the construction of the adjacency matrix of a MAG with its companion tuple, since it is used in the determination of the numerical representation of a composite vertex ($D(\mathbf{u},\tau)$). Therefore, the adjacency matrix of any given MAG is always presented with its companion tuple.

The adjacency matrix of a given MAG H is constructed by Algorithm 5, where $|\mathbb{V}(H)|$ is the number of composite vertices in H, which can be calculated using Equation (19), $D(\pi_o(e),\tau)$ and $D(\pi_d(e),\tau)$ are the numerical representation of the origin and destination composite vertices of edge $e \in E(H)$, respectively, as defined in Section 3.2.

Algorithm 5: Building $\mathbf{J}(H)$ from MAG H.

input : $H = (A, E)$
output: $\mathbf{J}(H), \tau(H)$

1 **AdjMatrix(**H**)**
2 $n \leftarrow |\mathbb{V}(H)|$
3 $T \leftarrow CompTuple(A(H))$ `// companion tuple`
4 $\mathbf{J}(H) \leftarrow n \times n$ matrix with all entries $= 0$
5 **for** *each* $e \in E(H)$ **do**
6 $\mathbf{u} \leftarrow D(\pi_o(e), T)$ `// numerical origin`
7 $\mathbf{v} \leftarrow D(\pi_d(e), T)$ `// numerical destination`
8 $\mathbf{J}(H)[\mathbf{u},\mathbf{v}] \leftarrow 1$
9 **end**
10 **return** $\mathbf{J}(H), T$

Considering that a sparse matrix with all entries 0 can be created in constant time, and that both functions *CompTuple* and D (see Algorithms 1 and 3) have time complexity $O(p)$, we conclude that Algorithm 5 has time complexity $O(p * |E(H)|)$, where p is the number of aspects of MAG H and $|E(H)|$ the number of edges.

As an example, the adjacency matrix of the MAG T is shown in Expression (21). This adjacency matrix $\mathbf{J}(T) \in \mathbb{R}^{18\times 18}$ has 324 entries, of which just 22 are non-zero.

$$\mathbf{J}(T) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{21}$$

It is important to note that the sequence of the columns and rows of $\mathbf{J}(T)$ is given by the numerical representation of the composite vertices. Thus, for instance, the 1 at row 2, column 8 represents the edge between the composite vertices with numerical representations 2 and 8, witch in turn represents the edge $(2, Bus, t_1, 2, Bus, t_2)$ of the MAG T. In this way, although $\mathbf{J}(T)$ is presented in matrix form, together with the companion tuple $\tau(T)$, it fully represents the MAG T, carrying the proper adjacency notion used to define transitive constructions, such as walks and paths on the MAG.

For an arbitrary MAG H, its main components graph $m(H)$ is obtained by removing the MAG's trivial components, as stated in Section 3.3. The matrix $\mathbf{J}(m(H))$ is then obtained with the use of the matrix $\mathbf{R}(H)$, presented in Section 3.3. $\mathbf{J}(m(H))$ is obtained as

$$\mathbf{J}(m(H)) = \mathbf{R}(H)^T \, \mathbf{J}(H) \, \mathbf{R}(H), \tag{22}$$

where $\mathbf{J}(m(H)) \in \mathbb{R}^{n-r\times n-r}$ is the adjacency matrix containing only the main components of the MAG.

It is also possible to obtain the adjacency matrix $\mathbf{J}(H)$ from $\mathbf{J}(m(H))$. This follows from the fact that on the adjacency matrix $\mathbf{J}(H)$ the rows and columns corresponding to trivial components are already zero. Therefore,

$$\mathbf{J}(H) = \mathbf{I}_m(H) \, \mathbf{J}(H) \, \mathbf{I}_m(H), \tag{23}$$

where $\mathbf{I}_m(H) = \mathbf{R}(H) \, \mathbf{R}(H)^T$. Then, we have that

$$\begin{aligned}
\mathbf{R}(H) \, \mathbf{J}(m(H)) \, \mathbf{R}(H)^T &= \mathbf{R}(H) \, \mathbf{R}(H)^T \, \mathbf{J}(H) \, \mathbf{R}(H) \, \mathbf{R}(H)^T \\
&= \mathbf{I}_m(H) \, \mathbf{J}(H) \, \mathbf{I}_m(H) \\
&= \mathbf{J}(H).
\end{aligned} \tag{24}$$

Expression (25) shows $\mathbf{J}(m(T))$, the adjacency matrix of $m(T)$. This matrix is obtained from the adjacency matrix $\mathbf{J}(T)$ by removing the rows and columns which represent the trivial components of the MAG T. In this case, the trivial components are the composite vertices with numerical representations $1, 6, 7, 12, 13$, and 18. This matrix is calculated as $\mathbf{J}(m(T)) = \mathbf{R}(T)^T \, \mathbf{J}(T) \, \mathbf{R}(T)$, so that

$$\mathbf{J}(m(T)) = \begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}. \tag{25}$$

In general, the adjacency matrices associated with a MAG are sparse, meaning that for an $n \times n$ adjacency matrix the number of non-zero entries of the matrix is of the order $O(n)$. Since the non-zero entries on the MAG adjacency matrix corresponds to the edges present on the MAG, the adjacency matrix being sparse means that the number of edges m on the MAG is of the same order of the number of composite vertices n, i.e., m is of order $O(n)$. Therefore, these matrices can be stored efficiently using sparse matrices representations, such as Compressed Sparse Column (CSC) or Compressed Sparse Row (CSR) [33], because such representations omit all entries of value zero. Assuming that the number of edges is larger than the number of composite vertices, these representations provide a space complexity of $O(m)$ for storing the MAG's adjacency matrices. Further, they also provide efficient matrix operations, which will be explored in the algorithms presented in Section 4.

3.5. Incidence Matrix

Given that every MAG is isomorphic to a directed graph, it follows that it can be represented by an incidence matrix (and its companion tuple). For any given MAG H, this matrix is constructed from the composite vertices $g(H)$ and the companion tuple $\tau(H)$, adopting the vertex sequence induced by the numerical representation presented in Section 3.2. The MAG's incidence matrix $\mathbf{C}(H) \in \mathbb{R}^{m \times n}$, where $m = |E(H)|$ is the number of edges in the MAG and $n = |\mathbb{V}(H)|$ is the number of composite vertices on the MAG, is defined then as

$$c_{e,\mathbf{u}} = \begin{cases} 1 & \text{if } \mathbf{u} = \pi_o(e), \\ -1 & \text{if } \mathbf{u} = \pi_d(e), \\ 0 & \text{otherwise,} \end{cases} \tag{26}$$

where $e \in E(g(H))$ is an edge in MAG H and $\mathbf{u} \in \mathbb{V}(H)$ is a composite vertex in MAG H. Here, the notation $c_{e,\mathbf{u}}$ is a shorthand for $c_{I_d(e),D(\mathbf{u},\tau)}$, where $I_d(e)$ is an numerical index for each edge and $D(\mathbf{u}, \tau)$ is the numerical representation of the composite vertex $\mathbf{u}$. Note that the use of the composite vertex numerical representation ties the incidence matrix to the MAG's companion tuple.

Although the sequence of the composite vertices is defined by each composite vertex numerical representation, the sequence used to present the MAG edges in the incidence matrix is not relevant. The incidence matrix of a directed graph has several well-known properties [34], among which, the property that the incidence matrix of a directed graph with k connected components has rank $n - k$, where n is the number of vertices of the graph. This property is useful for defining other matrices based on the incidence matrix.

For a given MAG H, the incidence matrix $\mathbf{C}(H)$ is built by Algorithm 6, where $D(\pi_o(e), T)$ and $D(\pi_d(e), T)$ are the numerical representation of the origin and destination composite vertices of edge $e \in E(H)$, respectively, as defined in Section 3.2, and $I_d(e)$ is a unique numerical index for the edge $e \in E(H)$, ranging from 1 to m. Considering that a sparse matrix with all entries 0 can be created in constant time, and that both functions *CompTuple* and D (see Algorithms 1 and 3) have time complexity of $O(p)$, we conclude that Algorithm 6 has time complexity of $O(p * |E(H)|)$, where p is the number of aspects of MAG H and $|E(H)|$ the number of edges.

Algorithm 6: Building $\mathbf{C}(H)$ from MAG H.

input : $H = (A, E)$
output: $\mathbf{C}(H), \tau(H)$

IncidMatrix(H)
- $n \leftarrow |\mathbb{V}(H)|$
- $m \leftarrow |E(H)|$
- $T \leftarrow CompTuple(A(H))$ `// companion tuple of H`
- $\mathbf{C}(H) \leftarrow m \times n$ matrix with all entries $= 0$
- **for** *each* $e \in E(H)$ **do**
  - $i \leftarrow I_d(e)$ `// index of edge e`
  - $\mathbf{u} \leftarrow D(\pi_o(e), T)$ `// numerical origin`
  - $\mathbf{v} \leftarrow D(\pi_d(e), T)$ `// numerical destination`
  - $\mathbf{C}(H)[i, \mathbf{u}] \leftarrow 1$
  - $\mathbf{C}(H)[i, \mathbf{v}] \leftarrow -1$
- **end**

return $\mathbf{C}(H), T$

Given the incidence matrix $\mathbf{C}(H)$ of a MAG H, it is possible to obtain the incidence matrix of the main components graph $\mathbf{C}(m(H))$ using the matrix $\mathbf{R}(H) \in \mathbb{R}^{n \times n-r}$ defined in Section 3.4. The incidence matrix of $m(H)$ is given by

$$\mathbf{C}(m(H)) = \mathbf{C}(H)\ \mathbf{R}(H). \tag{27}$$

Further, given the incidence matrix of the MAG's main components graph and the matrix $\mathbf{R}(H)$, it is possible to recover the MAG's incidence matrix, as

$$\begin{aligned} \mathbf{C}(m(H))\ \mathbf{R}(H)^T &= \mathbf{C}(H)\ \mathbf{R}(H)\ \mathbf{R}(H)^T \\ &= \mathbf{C}(H)\ \mathbf{I}_m(H) \\ &= \mathbf{C}(H). \end{aligned} \tag{28}$$

This is only possible because the columns of $\mathbf{C}(H)$, which are forced to $\mathbf{0}$ by the multiplication by $\mathbf{I}_m(H)$, were already $\mathbf{0}$, as the composite vertices represented by them have no edges incident to them.

The incidence matrix $\mathbf{C}(T)$ of the example MAG T is shown in Expression (29). The vertices (columns) sequence is determined by the vertices numerical representation, while the edge sequence remains unconstrained. The trivial components correspond to columns $1, 6, 7, 12, 13$, and 18, which have all entries with value 0.

$$\mathbf{C}(T) = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{29}$$

The main components incidence matrix $\mathbf{C}(m(T))$ is depicted in Expression (30). It is obtained from the matrix $\mathbf{C}(T)$ by removing the trivial components, as shown in Expression (28).

$$\mathbf{C}(m(T)) = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \end{bmatrix} \tag{30}$$

In general, the incidence matrices related to MAGs are sparse, and therefore can be efficiently stored using sparse matrices representations, such as CSC or CSR [33]. Assuming that the number of edges on the MAG is larger than the number of composite vertices, the use of these representation lead to a memory complexity of $O(m)$, where $m = |E(H)|$ is the number of edges on the MAG H.

4. MAG Algorithms

The MAG algorithms covered in this section are based on the MAG's adjacency matrix or on its adjacency list. Since in general we expect the adjacency matrix to be represented using sparse CSR ou CSC formats [33], it follows, due to the structure of the CSR and CSC formats, that the adjacency matrix and adjacency list can be seen as very closely related representations. The algorithms used in MAGs are directly derived from the basic well-known algorithms used with directed graphs [1,5,32,33]. In this section, we discuss algorithms for degree computation, BFS, and DFS in MAGs, since these algorithms are well-known and fairly simple, being thus convenient for the intended didactical purpose of this work. Furthermore, these basic algorithms also constitute building blocks for more sophisticated graph algorithms. In this sense, the purpose of this section is not to propose new algorithms, but to show how known algorithms may be adapted for application in MAGs. We remark that the presented BFS and DFS algorithms follow the same implementation shown in [5].

4.1. Auxiliary Matrices and Vectors

When operating upon a matrix representation, a few auxiliary matrices and vectors are necessary to express the desired operations. We now define these vectors, which are used on the remainder of this section. We denote $\mathbf{0}$ and $\mathbf{1}$ the column vectors with all entries equal to 0 and 1, respectively. In all cases, we assume the vectors have the dimension necessary for the operation where they are applied. When necessary to improve readability, we indicate the dimension by a sub-script as in $\mathbf{0}_n$ or $\mathbf{1}_n$.

Moreover, specially constructed matrices are used to build sub-determined algebraic algorithms for MAGs. These matrices provide reduction/aggregation operations needed for sub-determined algorithms. Although these matrices are specially constructed for the MAG and the sub-determination in question, they have distinct properties and can be constructed by a general algorithm. In fact, the construction of sub-determined algorithms relies on the use of functions to aggregate/reduce results according to the applied sub-determination. In some cases, this function can be as simple as just summing up values obtained in composite vertices, which are reduced to the same sub-determined vertex. However, depending on the algorithm being constructed, this aggregation may need a more elaborate function, which may not be expressed in terms of matrix multiplications.

Given a MAG H and a sub-determination ζ, the sub-determination matrix $\mathbf{M}_\zeta(H) \in \mathbb{R}^{n_\zeta \times n}$ is a rectangular matrix, where $n = |\mathbb{V}(H)|$ is the number of composite vertices of H and $n_\zeta = |\mathbb{V}_\zeta(H)|$ is the number of composite vertices of the sub-determination ζ applied to the MAG H. Since a

sub-determination is a (proper) subset of the aspects of a MAG, it follows that $n_\zeta \mid n$, i.e., the number of composite vertices of a MAG is a multiple of the number of composite vertices in any of its sub-determinations. Further, $\mathbf{M}_\zeta(H)$ has the property of having exactly one non-zero entry in each column, and the position of this entry is determined by the numerical value of the sub-determined composite vertex.

Algorithm 7 shows the construction of the sub-determination matrix $\mathbf{M}_\zeta(H)$ for a given MAG H and sub-determination ζ. The function D takes a composite vertex to its numerical representation and the function S_ζ takes a composite vertex to its sub-determined form, i.e., it drops the aspects not present in the sub-determination. To determine the time complexity of Algorithm 7, we consider that the count of composite vertices in line 3 is $O(|\mathbb{V}(H)|)$, the same is the case for the count on line 4, the construction of companion tuple at line 2 is $O(p)$, the construction of an empty sparse matrix at line 5 is $O(1)$, and, finally, the **for** loop initiated at line 6 is also $O(|\mathbb{V}(H)|)$. Since the number of aspects $p \ll |\mathbb{V}(H)|$, we conclude that the time complexity of Algorithm 7 is $O(|\mathbb{V}(H)|)$.

Algorithm 7: Construction of $\mathbf{M}_\zeta$.

input : $\tau(H)$ and ζ
output: $\mathbf{M}_\zeta(H)$

1 **SubDetMatrix(**$\tau(H)$**,** ζ**)**
2 $T_\zeta = SubCompTuple(\tau(H), \zeta)$ `// ζ sub-determined companion tuple`
3 $n \leftarrow |\mathbb{V}(H)|$
4 $z \leftarrow |\mathbb{V}_\zeta(H)|$
5 $\mathbf{M}_\zeta(H) \leftarrow z \times n$ sparse matrix
6 **for** $j \leftarrow 1$ ***to*** n **do**
7 $\mathbf{u} \leftarrow D^{-1}(j, \tau(H))$ `// numeric tuple form of j`
8 $i \leftarrow D(\mathbf{u}, T_\zeta)$ `// sub-determined numerical representation`
9 $\mathbf{M}_\zeta(H)[i, j] \leftarrow 1$
10 **end**
11 **return** $\mathbf{M}_\zeta(H)$

For instance, consider the example MAG T and a sub-determination $\zeta_t = 011_2$, which drops the third aspect of T. The aspect dropped is the aspect of time instants and, therefore, the two aspects present in ζ_t are location and transit layers. Since in T there are 3 locations and 2 transit layers, it follows that $|\mathbb{V}_{\zeta t}(T)| = 6$. Hence, $\mathbf{M}_{\zeta t}(T) \in \mathbb{R}^{6\times 18}$ constructed according to Algorithm 7 is given by

$$\mathbf{M}_{\zeta t}(T) = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}. \tag{31}$$

As a further example, consider the MAG T and a sub-determination $\zeta_T = 100_2$, which drops the location and transit layer aspects, leaving only the time instants aspects. Since there are 3 time instants in T, it follows that $M_{\zeta T}(T) \in \mathbb{R}^{3\times 18}$ is

$$\mathbf{M}_{\zeta T}(T) = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix}. \tag{32}$$

Note that in these cases the multiplication by the sub-determination matrices performs the sum of the distinct composite vertices that are reduced to a same sub-determined vertex. For instance, given the sub-determination $\zeta_t = 011_2$, the matrix $\mathbf{M}_{\zeta t}(T)$ is used to aggregate values found in 3 composite

vertices into a single sub-determined vertex. The aggregation function in this case is a simple sum. The same is done by the matrix $\mathbf{M}_{\zeta_T}(T)$ for the sub-determination $\zeta_T = 100_2$, where in this case each sub-determined result is the sum of values obtained for 6 composite vertices.

4.2. Universality of Matrix Algorithms

In this section, we show that every function that can be obtained from a MAG to a given co-domain set can also be obtained from a matrix representation of the MAG. Here the set $\mathbb{H}$ is the quotient set of finite MAGs under isomorphism defined in Section 2.1. Note that a permutation σ of a given adjacency matrix $\mathbf{J}(H)$, together with the function D_σ, represents the same MAG H as $\mathbf{J}(H)$, so that permutations of adjacency matrices are isomorphic. Thus, we have the set $\mathbb{J}$, which is a quotient set of pairs $(\mathbf{J}_\sigma, D_\sigma)$ of adjacency matrices and association functions D, under adjacency matrix permutations. Therefore, an element of $\mathbb{J}$ is an equivalence class of adjacency matrices and D functions. Since we consider the pair $(\mathbf{J}(H), \tau(H))$ as the canonical adjacency matrix representation of the MAG H, we assign this pair as the class representative of the MAG H in $\mathbb{J}$.

Theorem 1. *The adjacency matrix* $\mathbf{J}(H)$ *and companion tuple* $\tau(H)$ *obtained from the MAG* H *by Algorithm 5 are isomorphic to the MAG* H.

Proof. We show that Algorithm 5 can be seen as a function that takes a given MAG H to its adjacency matrix and companion tuple, and that this function preserves the adjacency structure of the original MAG. Further, we show that, from the adjacency matrix $\mathbf{J}(H)$ and companion tuple $\tau(H)$, we can construct a MAG $\hat{H}$ that is isomorphic to MAG H.

- $\Longrightarrow$
  Given the sets $\mathbb{H}$ and $\mathbb{J}$, Algorithm 5 can be seen as a function

$$\begin{aligned} \Upsilon : \mathbb{H} &\to \mathbb{J} \\ H &\mapsto (\mathbf{J}(H), \tau(H)). \end{aligned} \tag{33}$$

  Considering the loop depicted at lines 5 to 9 in Algorithm 5, it can be seen that every edge $e \in E(H)$ is converted in a pair of composite vertices ($\mathbf{u}$ and $\mathbf{v}$) and then represented as an edge on the adjacency matrix $\mathbf{J}(H)$. Therefore, if the composite vertices $\mathbf{u}$ and $\mathbf{v}$ are adjacent in MAG H, then a entry 1 is present at the intersection of row $D(\mathbf{u}, \tau(H))$ and column $D(\mathbf{v}, \tau(H))$ of $\mathbf{J}(H)$, indicating the corresponding adjacency in the matrix. Hence, the adjacency structure of the MAG H is preserved by the function Υ.
- $\Longleftarrow$
  Given the adjacency matrix $\mathbf{J}(H)$ and companion tuple $\tau(H)$, we construct MAG $\hat{H}$, which we then show to be isomorphic to the MAG H. We obtain $A(\hat{H})$ from $\tau(H)$ by constructing a list $A(\hat{H})$ with $p = |\tau(H)|$ elements, in which every element i of this list is a set such that $|A(\hat{H})[i]| = \tau[i]$. Without loss of generality, we can assume that the elements of each aspect $A(\hat{H})[i]$ are natural numbers ranging from 1 to $\tau[i]$. We then construct the edge set $E(\hat{H})$, by starting with an empty set and then inserting an edge for each entry with value 1 in $\mathbf{J}(H)$. For constructing each of these edges, we take an entry of value 1, make r equal to its row number, and c equal to its column number. We then build the edge $e = (D^{-1}(r, \tau(H)), D^{-1}(c, \tau(H))$, which is a tuple of length $2p$, where the first p entries correspond to the origin composite vertex and the last p entries correspond to the destination composite vertex of the edge. Note that function D^{-1} simply retrieves the original composite vertex entries from the row and column numbers of the adjacency matrix. Further, we construct the set $\mathbb{V}(\hat{H})$ of composite vertices, which is the cartesian product of the sets in $A(\hat{H})$, so that

$$\mathbb{V}(\hat{H}) = \bigtimes_{n=1}^{p} A(\hat{H})[n], \tag{34}$$

where $p = |\tau(H)|$ is the number of aspects in the MAG H.

We now show that the MAG $\hat{H}$, constructed from $\mathbf{J}(H)$ and $\tau(H)$, is isomorphic to the original MAG H. Note that by construction of $\hat{H}$ we have that $|E(\hat{H})| = |E(H)|$; $|A(\hat{H})| = |A(H)| = p$; for $1 \leqslant i \leqslant p, |A(\hat{H})[i]| = |A(H)[i]|$; $|\mathbb{V}(\hat{H})| = |\mathbb{V}(H)|$; and $\tau(\hat{H}) = \tau(H)$.

Since $|\mathbb{V}(H)| = |\mathbb{V}(\hat{H})|$, we know that there is a bijective function from $\mathbb{V}(\hat{H})$ to $\mathbb{V}(H)$. Further, we also have the bijective function D, which takes a composite vertex into a natural number, assigning a unique and distinct natural number to each element of $\mathbb{V}(H)$ and $\mathbb{V}(\hat{H})$. Moreover, since $\tau(\hat{H}) = \tau(H)$ and by construction of D, we have that the range of D for $\mathbb{V}(H)$ and $\mathbb{V}(\hat{H})$ is the same, i.e., $D(\mathbb{V}(H), \tau(H)) = D(\mathbb{V}(\hat{H}), \tau(\hat{H}))$. From this, we conclude that, for every composite vertex $\mathbf{u} \in \mathbb{V}(H)$, there is one unique composite vertex $\hat{\mathbf{u}} \in \mathbb{V}(\hat{H})$ such that $D(\mathbf{u}, \tau(H)) = D(\hat{\mathbf{u}}, \tau(\hat{H}))$. We thus define the bijective function

$$\begin{aligned} f : \mathbb{V}(\hat{H}) &\to \mathbb{V}(H) \\ \hat{\mathbf{u}} &\mapsto \mathbf{u}, \text{ such that } D(\hat{\mathbf{u}}, \tau(\hat{H})) = D(\mathbf{u}, \tau(H)). \end{aligned} \tag{35}$$

As the function f is bijective, for every edge $\hat{e} \in \hat{H}$, we have an edge $e = (f(\pi_o(\hat{e})), f(\pi_d(\hat{e}))) \in E(H)$, and also, for every edge $e \in E(H)$, we have the corresponding edge $\hat{e} = (f^{-1}(\pi_o(e)), f^{-1}(\pi_d(e))) \in E(\hat{H})$. This fulfils the conditions for isomorphism between $\hat{H}$ and H.

Since $\mathbb{H}$ is a quotient set under the MAG isomorphism relation and $\hat{H}$ is isomorphic to H, it follows that $\hat{H}$ and H correspond to the same element in $\mathbb{H}$, making the function Y bijective. Also, since each entry with value 1 in the adjacency matrix $\mathbf{J}(H)$ corresponds to an edge in the MAG H, it follows that Y^{-1} also preserves the MAGs adjacency structure, establishing the isomorphism relation as desired.

□

Theorem 2. *Every function that can be obtained from a MAG to a given co-domain set can also be obtained from a matrix representation of the MAG.*

Proof. Consider the diagram depicted in Figure 5. In this figure, $\mathbb{H}$ is the set of all MAGs (up to isomorphism), $\mathbb{J}$ is the set of pairs of adjacency matrices and companion tuples (up to permutation), F is an arbitrary function from $\mathbb{H}$ to $\mathbb{X}$, where $\mathbb{X}$ is a codomain consistent with the definition of function F, and I is the identity function in $\mathbb{X}$. Since the function F is arbitrary, it can represent any function or algorithm, such as searches or centrality computations, which take MAGs to a result expected from this function.

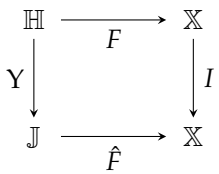


Figure 5. Commutative diagram.

As both functions Y (Equation (33) in Theorem 1) and I represent isomorphisms, it follows that the depicted diagram commutes, so that for every function $F : \mathbb{H} \to \mathbb{X}$ there is a function $\hat{F} : \mathbb{J} \to \mathbb{X}$, which produces the same result. □

As a consequence of Theorem 2, it follows that, from the adjacency matrix and companion tuple of a MAG, one can obtain any possible outcome that can be obtained from a MAG or from any other representation equivalent to it, such as high order tensors, as those presented in recent related

works [15,35,36]. Therefore, Theorem 2 actually shows that any possible MAG function (algorithm) can be obtained from a matrix-based representation of a MAG. This is an important theoretical result since it paves the way for adapting well-known graph algorithms for application in MAGs. In the next subsections, we show examples of basic algorithms and functions that can be obtained from the matrix-based representation of a MAG.

4.3. Degree

The definition of degree in a traditional graph stems from the number of edges incident to a given vertex. This concept can be generalized for MAGs, so that degrees can be defined for composite vertices, sub-determinations, or elements of a given aspect. Further, since MAG edges are considered to be directed, the degrees are also divided into out-degree and in-degree. In this section, we present algorithms for calculating these distinct degree definitions.

4.3.1. Degree of Composite Vertices

The degree of composite vertices of a given MAG H can be obtained directly from its composite vertices representation, $g(H)$. Since the composite vertices representation is a traditional directed graph isomorphic to the MAG H, it follows that the degree determination is done with the traditional algorithm for directed graphs with minor changes. For a given MAG H and its companion tuple $\tau(H)$, the degrees of the composite vertices can be determined by Algorithm 8, where $D(\pi_o(e), \tau(H))$ and $D(\pi_d(e), \tau(H))$ stand for the numerical representation of the origin and destination composite vertices of edge $e \in E(H)$, as defined in Section 3.2.

Algorithm 8: Determination of the degree of composite vertices.

input : $H = (A, E)$
output: *indegree*, *outdegrees*

Degree(H**)**
  $n \leftarrow |\mathbb{V}(H)|$
  $T \leftarrow CompTuple(A(H))$ `// companion tuple of H, i.e., τ(H)`
  indegree ← vector of n integers, all 0
  outdegree ← vector of n integers, all 0
  for *each* $e \in E(H)$ **do**
    $o \leftarrow D(\pi_o(e), T)$ `// numerical origin`
    $d \leftarrow D(\pi_d(e), T)$ `// numerical destination`
    $indegree[d] \leftarrow indegree[d] + 1$
    $outdegree[o] \leftarrow outdegree[o] + 1$
  end
return *indegree*, *outdegree*

Another way for calculating the degrees of the composite vertices is computing it algebraically from the adjacency matrix of the MAG, as given by

$$indegree = \mathbf{J}(H)^T \mathbf{1}, \tag{36}$$

and

$$outdegree = \mathbf{J}(H) \mathbf{1}. \tag{37}$$

Further, the total degree of the composite vertices can be obtained by summing up their indegrees and outdegrees.

To determine the time complexity of Algorithm 8, we consider that lines 2, 4, and 5 have each time complexity $O(|\mathbb{V}(H)|)$, the determination of the companion tuple at line 3 has complexity $O(p)$, where p is the number of aspects of the MAG, so that $p \ll |\mathbb{V}(H)|$. Finally, since the determination of the numerical representation of vertices has complexity $O(p)$, we have that the **for** loop initiated at line 6 has complexity $O(p * |E(H)|)$, so that the time complexity of Algorithm 8 is $O(|\mathbb{V}(H)| + p * |E(H)|)$. If we consider that in a given case the order of the MAG does not vary, so that p is a constant, then the algorithm's time complexity is $O(|\mathbb{V}(H)| + |E(H)|)$.

In the case of the example MAG T (Figure 3), whose companion tuple is $\tau = (3, 2, 3)$, it can be seen that the composite vertex $(2, Bus, t1)$ has outdegree 3 and indegree 1, while the composite vertex $(1, Subway, t2)$ has outdegree 2 and indegree 2. Since $D((2, Bus, t1), \tau) = 2$ and $D((1, Subway, t2), \tau) = 10$, it follows that $indegree[2] = 1$, $outdegree[2] = 3$, $indegree[10] = 2$ and $outdegree[10] = 2$.

4.3.2. Degree of Sub-Determined Vertices

We can determine the degree for sub-determined composite vertices in a similar way to the degree of composite vertices. Given a MAG H and a sub-determination ζ, the degree of the sub-determined composite vertices can be obtained by Algorithm 9, where $|\mathbb{V}_\zeta(H)|$ is the number of ζ sub-determined composite vertices on MAG H, S_ζ is the function that takes a composite vertex to its sub-determined form, and D_ζ is the function that takes the sub-determined composite vertex to its numerical representation. It can be seen that the time complexity of Algorithm 9 is the same as the time complexity of Algorithm 8.

Algorithm 9: Sub-determined degree.

input : $H = (A, E)$, and ζ
output: *indegree*, *outdegree*

```
1  SubDetDegree(H, ζ)
2      n ← |V_ζ(H)|
3      T ← CompTuple(A(H))                    // companion tuple of H, i.e., τ(H)
4      T_ζ ← τ_ζ(H)                           // ζ sub-determined companion tuple
5      indegree ← vector of n integers, all 0
6      outdegree ← vector of n integers, all 0
7      for each e ∈ E(H) do
8          o ← D(π_o(e), T_ζ)                 // numerical sub-determined origin
9          d ← D(π_d(e), T_ζ)                 // numerical sub-determined destination
10         indegree[d] ← indegree[d] + 1
11         outdegree[o] ← outdegree[o] + 1
12     end
13 return indegree, outdegree
```

It is important to note that two distinct composite vertices may have the same sub-determined form. This happens when the two composite vertices differ only on aspects which are dropped by the sub-determination. In this case, the degree of each of these composite vertices is summed for obtain the sub-determined degree. From this, it can also be seen that some edges in the sub-determined form may become self-loops. The degrees calculated by Algorithm 9 include the self-loop edges. This algorithm can be modified to count the self-loops separately, as shown in Algorithm 10. This algorithm is similar to Algorithm 8 and has the same time complexity.

The sub-determined composite vertices degree can also be determined algebraically with

$$indegree = \mathbf{M}_\zeta(H)\, \mathbf{J}(H)^T\, \mathbf{1}, \tag{38}$$

and

$$outdegree = \mathbf{M}_\zeta(H)\ \mathbf{J}(H)\ \mathbf{1}, \tag{39}$$

where $\mathbf{M}_\zeta(H)$ is the sub-determination matrix and $\mathbf{1}$ is the all 1s column vector, both defined in Section 4.1. Note that the multiplication by $\mathbf{M}_\zeta(H)$ adds the degrees of the composite vertices that are collapsed to the same sub-determined vertex.

Algorithm 10: Sub-determined degree, separating self-loops

input : $H = (A, E)$, and ζ
output: *indegree*, *outdegree*, *selfdegree*

SubDetDegreeSepLoops(H, ζ**)**
  $n \leftarrow |\mathbb{V}_\zeta(H)|$
  $T \leftarrow CompTuple(A(H))$ `// companion tuple of H, i.e., τ(H)`
  $T_\zeta \leftarrow \tau_\zeta(H)$ `// ζ sub-determined companion tuple`
  indegree ← vector of n integers, all 0
  outdegree ← vector of n integers, all 0
  selfdegree ← vector of n integers, all 0
  for *each* $e \in E(H)$ **do**
    $o \leftarrow D(\pi_o(e), T_\zeta)$ `// numerical sub-determined origin`
    $d \leftarrow D(\pi_d(e), T_\zeta)$ `// numerical sub-determined destination`
    if $d \neq o$ **then**
      $indegree[d] \leftarrow indegree[d] + 1$
      $outdegree[o] \leftarrow outdegree[o] + 1$
    end
    else
      $selfdegree[o] \leftarrow selfdegree[o] + 1$
    end
  end
return *indegree*, *outdegree*, *selfdegree*

The degrees calculated by Equations (38) and (39) include the self-loop edges. To obtain the separate self-loop degrees, first note that

$$\mathbf{M}_\zeta(H)\ \mathbf{J}(H)\ \mathbf{1}_n = \mathbf{M}_\zeta(H)\ \mathbf{J}(H)\ \mathbf{M}_\zeta(H)^T\ \mathbf{1}_m. \tag{40}$$

This follows from the fact that

$$\mathbf{M}_\zeta(H)^T\ \mathbf{1}_m = \mathbf{1}_n, \tag{41}$$

since $\mathbf{M}_\zeta(H)^T$ is a $n \times m$ rectangular matrix and has the property that each row has exactly one non-zero entry of value 1.

Furthermore, note that the matrix $\mathbf{M}_\zeta(H)\ \mathbf{J}(H)\ \mathbf{M}_\zeta(H)^T$ is the adjacency matrix of the sub-determined MAG H_ζ. Since the composite vertices representation of a sub-determined MAG is a multigraph, each non-zero entry shows the number of superposed edges in the sub-determination. Therefore, the main diagonal of $\mathbf{M}_\zeta(H)\ \mathbf{J}(H)\ \mathbf{M}_\zeta(H)^T$ has the self-loop degree of each vertex. Hence,

$$selfdegree = Diag(\mathbf{M}_\zeta(H)\ \mathbf{J}(H)\ \mathbf{M}_\zeta(H)^T). \tag{42}$$

For example, consider the example MAG T (Figure 3) and the sub-determination $\zeta_t = 011_2$ defined in Section 4.1. We have that

$$indegree = \mathbf{M}_{\zeta t}(T)\,\mathbf{J}(T)^T\,\mathbf{1} = \begin{bmatrix} 0 \\ 7 \\ 4 \\ 4 \\ 7 \\ 0 \end{bmatrix}, \tag{43}$$

$$outdegree = \mathbf{M}_{\zeta t}(T)\,\mathbf{J}(T)\,\mathbf{1} = \begin{bmatrix} 0 \\ 7 \\ 4 \\ 4 \\ 7 \\ 0 \end{bmatrix}, \tag{44}$$

$$\mathbf{M}_{\zeta t}(T)\,\mathbf{J}(T)\,\mathbf{M}_{\zeta t}(T)^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 0 & 3 & 0 \\ 0 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 0 \\ 0 & 3 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{45}$$

and

$$selfdegree = Diag(\mathbf{M}_{\zeta t}(T)\,\mathbf{J}(T)\,\mathbf{M}_{\zeta t}(T)^T) = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 2 \\ 2 \\ 0 \end{bmatrix}. \tag{46}$$

This means that, for instance, the sub-determined composite vertex $(2, Subway)$ has outdegree 7, indegree 7, and 2 self-loops. This sub-determination corresponds to the aggregation of all 3 time instants, which means that the edges in which only the time instant changes become self-loops. These edges are shown in red (dotted) in Figure 3. Note that $\tau_{\zeta_t} = (2, 3)$, so that $D((2, Subway), (2, 3)) = 2$, making it correspond to the second element of the degree column vector.

4.3.3. Single Aspect Degree

The single aspect degree is a particular case of sub-determined degree in which the sub-determination applied is such that only a single aspect remains. Therefore, the determination of single aspect degrees is done in the same way presented in Section 4.3.2.

We, however, present an additional example illustrating the time instant degree, which is obtained by the sub-determination $\zeta_T = 100_2$ defined in Section 4.1. This sub-determination has only the third aspect of the MAG T (Figure 3), which corresponds to the three time instants present on MAG T. In this case, we have that

$$indegree = \mathbf{M}_{\zeta T}(T)\,\mathbf{J}(T)^T\,\mathbf{1} = \begin{bmatrix} 2 \\ 10 \\ 10 \end{bmatrix}, \tag{47}$$

$$outdegree = \mathbf{M}_{\zeta T}(T)\,\mathbf{J}(T)\,\mathbf{1} = \begin{bmatrix} 10 \\ 10 \\ 2 \end{bmatrix}, \tag{48}$$

$$\mathbf{M}_{\zeta T}(T)\,\mathbf{J}(T)\,\mathbf{M}_{\zeta T}(T)^T = \begin{bmatrix} 2 & 8 & 0 \\ 0 & 2 & 8 \\ 0 & 0 & 2 \end{bmatrix}, \tag{49}$$

and

$$selfdegree = Diag(\mathbf{M}_{\zeta T}(T)\,\mathbf{J}(T)\,\mathbf{M}_{\zeta T}(T)^T) = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}. \tag{50}$$

Therefore, we have that $\tau_{\zeta_T} = (3)$, so that $D((t_1), (3)) = 1$, $D((t_2), (3)) = 2$, and $D((t_3), (3)) = 3$. Considering the composite vertices representation of MAG T, depicted in Figure 3, it can be seen

that each time instant has 2 self-loop edges (in blue-dashed), which is consistent with Equation (50). Further, there are 8 edges from t_1 to t_2 (in red-dotted and black) and 8 edges from t_2 to t_3. This is consistent with the adjacency matrix shown in Equation (49). Further, the indegrees and outdegrees of each time instant are consistent with Equations (47) and (48).

4.4. Breadth-First Search (BFS)

The Breadth-First Search (BFS) is an important graph algorithm that can be seen as a primitive for building many other algorithms [5]. The goal of this section is to illustrate how the BFS algorithm can be adapted for being used in MAGs, both in its full composite vertices representation and in its sub-determined forms. In the not sub-determined form, the adaptation is very simple, since the composite vertices representation of a MAG is a directed graph. In this case, all that is needed is to convert the composite vertices representation from its tuple to numerical form, and then apply the traditional BFS algorithm. The adaptation to the sub-determined forms also does not require major changes on the algorithm. As with many graph algorithms, BFS can be expressed in combinational or in algebraic forms, which are presented in the following related subsections.

4.4.1. BFS for Composite Vertices

The non sub-determined BFS in its combinational form is constructed directly upon the MAG's adjacency matrix, $\mathbf{J}(H)$.

Algorithm 11: BFS for composite vertices.

input : $\mathbf{J}(H)$, $\tau(H)$, and $\mathbf{s} \in \mathbb{V}(H)$
output: *vertices*, *distance*, *pred*

BFS($\mathbf{J}(H), \tau(H), \mathbf{s}$)
  $n \leftarrow |\mathbb{V}(H)|$
  vertices $\leftarrow$ vector of n integers, all 0
  distance $\leftarrow$ vector of n integers, all ∞
  pred $\leftarrow$ vector of n integers, all *Nil*
  color $\leftarrow$ vector of n integers, all 0 `// set all vertices to white (unvisited)`
  $Q \leftarrow$ empty queue
  $vertices[D(\mathbf{s}, \tau(H))] \leftarrow 1$
  $color[D(\mathbf{s}, \tau(H))] \leftarrow 1$
  $distance[D(\mathbf{s}, \tau(H))] \leftarrow 0$
  $Enqueue(Q, D(\mathbf{s}, \tau(H)))$
  while Q *not empty* **do**
    $u \leftarrow head[Q]$
    for *each v successor of u* **do**
      if $color[v] = 0$ **then**
        $color[v] \leftarrow 1$ `// set vertex v to gray (visited)`
        $vertices[v] \leftarrow 1$
        $distance[v] \leftarrow distance[u] + 1$
        $pred[v] \leftarrow u$
        $Enqueue(Q, v)$
      end
    end
    $Dequeue(Q)$
    $color[u] \leftarrow 2$ `// set vertex u to black (closed)`
  end
return *vertices*, *distance*, *pred*

Considering Algorithm 11 and the standard form of the BFS algorithm encountered in [5], it can be seen that the difference is that the starting composite vertex **s** has to be transformed from its tuple representation to its numerical representation, as shown in lines 8, 9, and 10 of Algorithm 11. Therefore, from the analysis provided in [5], we can conclude that the time complexity of Algorithm 11 is $O(|\mathbb{V}(H)| + |E(H)|)$.

BFS is also closely related to matrix multiplication. This stems from the well-known property of the powers of the adjacency matrix, in which the (i, j) entry of the n-th power of the adjacency matrix shows the number of existing walks of length n from vertex i to vertex j [33]. From this, we could think that for a given MAG H, the series

$$\mathbf{B} = \sum_{i=0}^{\infty} \mathbf{J}(H)^i = \mathbf{I} + \mathbf{J}(H) + \mathbf{J}(H)^2 + \mathbf{J}(H)^3 + \mathbf{J}(H)^4 + \dots \tag{51}$$

would produce a matrix $\mathbf{B}$, such that the entry $\mathbf{B}_{i,j}$ indicates the number of walks of any length from vertex i to vertex j. This is indeed the case when H happens to be an acyclic MAG, making $\mathbf{J}(H)$ a nilpotent matrix.

The existence of cycles in H makes that, for some vertices, there will exist walks of arbitrary length connecting them (namely, the cycles), making the series of Equation (51) divergent. However, since the objective is not to know the number of walks between each pair of vertices, but simply to know which vertices are reachable from each other (i.e., there is at least a path between them), this technical problem can be solved by multiplying the adjacency matrix $\mathbf{J}(H)$ by a scalar ρ_H, such that

$$\rho_H < \frac{1}{\rho(\mathbf{J}(H))}, \tag{52}$$

where $\rho(\mathbf{J}(H))$ is the spectral radius of the matrix $\mathbf{J}(H)$. This leads to the matrix

$$\mathbf{J}_\rho(H) = \rho_H \, \mathbf{J}(H), \tag{53}$$

so that the spectral radius of the matrix $\mathbf{J}_\rho(H) < 1$. This results that Equation (51) constructed with the matrix $\mathbf{J}_\rho(H)$ converges. Since the convergence of the series is assured, Equation (51) can be re-expressed as

$$\mathbf{B} = (I - \mathbf{J}_\rho(H))^{-1}. \tag{54}$$

The matrix $\mathbf{B}$ defined in Equation (54) has the property that, for any given composite vertex $\mathbf{v} \in \mathbb{V}(H)$, the row $D(\mathbf{v})$ of $\mathbf{B}$ has non-zero entries in every column that corresponds to a composite vertex $\mathbf{u} \in \mathbb{V}(H)$, such that $\mathbf{u}$ is reachable from $\mathbf{v}$. Hence, for a given composite vertex $\mathbf{v}$, the row $D(\mathbf{v})$ corresponds to the result of a BFS started at that composite vertex. Although the matrix $\mathbf{B}$ carries the BFS of all composite vertices of the MAG H, it is important to note that this matrix may not be sparse, which for large MAGs can lead to difficulties in memory allocation. To avoid such difficulties, it is also possible to express a BFS for a single composite vertex $\mathbf{v}$ as

$$\mathbf{B} = r_\mathbf{v}\mathbf{I} + r_\mathbf{v}\mathbf{J}_\rho(H) + r_\mathbf{v}\mathbf{J}_\rho(H)^2 + r_\mathbf{v}\mathbf{J}_\rho(H)^3 + r_\mathbf{v}\mathbf{J}_\rho(H)^4 + \dots, \tag{55}$$

where $r_\mathbf{v}$ is the row vector with n entries for which all entries except $D(\mathbf{v}, \tau)$ are 0 and the entry $D(\mathbf{v}, \tau)$ is 1.

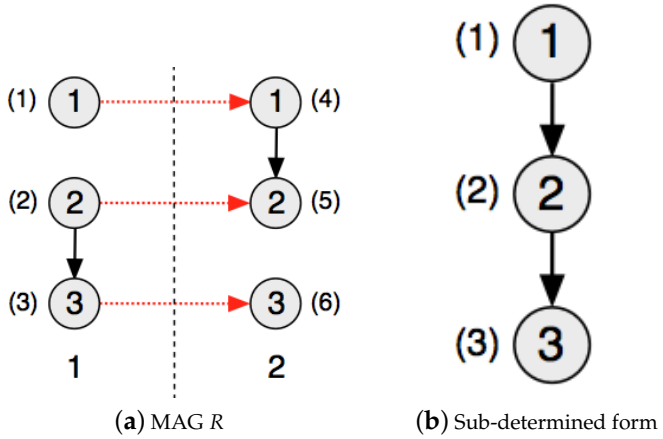
Considering the example MAG T, shown in Figure 3, the result of the BFS using Algorithm 11 for the composite vertex $(2, Bus, t_1)$, whose numerical representation is $D((2, Bus, t_1), (3, 2, 3)) = 2$, is

$$\begin{aligned} vertices &= [2, 5, 8, 9, 10, 11, 14, 15, 16, 17] \\ distances &= [\infty, 0, \infty, \infty, 1, \infty, \infty, 1, 1, 2, 2, \infty, \infty, 2, 2, 3, 3, \infty] \\ pred &= [Nil, Nil, Nil, Nil, 2, Nil, Nil, 2, 2, 5, 5, Nil, Nil, 8, 8, 10, 10, Nil], \end{aligned} \tag{56}$$

where the list *vertices* shows the composite vertices accessible from $(2, Bus, t_1)$, which in this example represent all locations, transit modals, and time instants reachable from this initial point. The list *distances* carries the distances in hops from the initial composite vertex $(2, Bus, t_1)$ to all possible destinations (with ∞ meaning that a destination is not reachable). The list *pred* shows the predecessors of each composite vertex, making possible to construct a BFS tree.

4.4.2. Sub-Determined BFS

It is possible to obtain a sub-determined form of the BFS algorithm for MAGs. It is important, however, to realize that this sub-determined BFS algorithm is not equivalent to applying the BFS algorithms presented in Section 4.4.1 to a sub-determined MAG. A sub-determination is a generalization of the idea of aggregating multilayer and time-varying graphs, as shown in Section 2.2. As with the aggregation process, the sub-determination of a MAG can cause the presence of paths and walks on the sub-determined MAG that do not actually exist on the original MAG. To illustrate this, we present Figure 6a,b, which show a small two aspects MAG and its sub-determined form, obtained by the sub-determination $\zeta_R = 01_2$. First, note that, in the MAG R shown in Figure 6a, there is no path originating from the composite vertices $(1,1)$ or $(1,2)$ to the composite vertices $(3,1)$ or $(3,2)$. Nevertheless, in Figure 6b, there is a path connecting the sub-determined vertex (1) to the sub-determined vertex (3), even though such connection is not possible on the original MAG shown in Figure 6a. Therefore, to obtain the proper result, the sub-determined BFS should not be evaluated directly using the sub-determined MAG.



(**a**) MAG R (**b**) Sub-determined form

Figure 6. MAG R and its sub-determined form.

Such a case can be seen algebraically by noting that given a MAG H and a sub-determination ζ, in general

$$\mathbf{M}_\zeta(H) \left(\sum_{i=0}^{\infty} \mathbf{J}_\rho(H)^i \right) \mathbf{M}_\zeta(H)^T \neq \sum_{i=0}^{\infty} \left(\mathbf{M}_\zeta(H)\, \mathbf{J}_\rho(H)\, \mathbf{M}_\zeta(H)^T \right)^i . \tag{57}$$

To see that the Inequality (57) holds, note that an arbitrary power of the matrix $\mathbf{M}_\zeta(H)\, \mathbf{J}_\rho(H)\, \mathbf{M}_\zeta(H)^T$ is given by

$$\left(\mathbf{M}_\zeta(H) \mathbf{J}_\rho(H) \mathbf{M}_\zeta(H)^T \right)^n = \underbrace{\mathbf{M}_\zeta(H) \mathbf{J}_\rho(H) \mathbf{M}_\zeta(H)^T\, \mathbf{M}_\zeta(H) \mathbf{J}_\rho(H) \mathbf{M}_\zeta(H)^T \ldots}_{\text{n times}}, \tag{58}$$

where $\left(\mathbf{M}_\zeta(H)\, \mathbf{J}_\rho(H)\, \mathbf{M}_\zeta(H)^T \right)$ is multiplied n times. Note, however, that

$$\mathbf{M}_\zeta(H)^T\, \mathbf{M}_\zeta(H) \neq \mathbf{I}_n, \tag{59}$$

since $\mathbf{M}_{\zeta}(H) \in \mathbb{R}^{m \times n}$ is a retangular matrix and $m < n$, so that the rank of the matrix $\mathbf{M}_{\zeta}(H)^T \, \mathbf{M}_{\zeta}(H)$ is less or equal to m, while the rank of the identity $\mathbf{I}_n$ is $n > m$. Since Inequality (59) holds, so does the Inequality (57).

Here, the left hand side of the Inequality (57) corresponds to the sub-determination of the BFS calculated for the MAG H, while the right hand side corresponds to the BFS calculated for the sub-determined MAG H_{ζ}.

In the case of the MAG R, shown in Figure 6a, we have that the sub-determination is given by $\zeta_R = 01_2$ and the adjacency and sub-determination matrices are

$$\mathbf{J}(R) = \begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{60}$$

$$\mathbf{M}_{\zeta_R}(R) = \begin{bmatrix} \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} \end{bmatrix}, \tag{61}$$

and

$$\mathbf{J}_{\zeta_R}(R) = \begin{bmatrix} 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 \end{bmatrix}. \tag{62}$$

Therefore, we have that

$$\mathbf{M}_{\zeta_R}(R) \left(\sum_{i=0}^{\infty} \mathbf{J}_{\rho}(R)^i \right) \mathbf{M}_{\zeta_R}(R)^T = \begin{bmatrix} 3 & 2 & \mathbf{0} \\ 0 & 3 & 2 \\ 0 & 0 & 3 \end{bmatrix}, \tag{63}$$

while

$$\sum_{i=0}^{\infty} \left(\mathbf{M}_{\zeta_R}(R) \, \mathbf{J}_{\rho}(R) \, \mathbf{M}_{\zeta_R}(R)^T \right)^i = \begin{bmatrix} 2 & 2 & \mathbf{2} \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{bmatrix} \tag{64}$$

and

$$\left(\mathbf{I}_3 - \mathbf{J}_{\zeta_R}(R) \right)^{-1} = \begin{bmatrix} 1 & 1 & \mathbf{1} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{65}$$

Remembering that the entries of the matrices in Equations (63)–(65) are to be considered only as zero or non-zero, it can be seen that the matrix at Equation (63) has a 0 at entry $(1,3)$, while the matrices at Equations (64) and (65) have a non-zero entry at this same position. This illustrates the situation in which a BFS is done on the sub-determined (aggregated) MAG, as in Equations (64) and (65), i.e., paths that are not present on the original MAG can appear on the sub-determined form, potentially altering the results obtained by algorithms applied to it.

For instance, considering the MAG T, depicted in Figure 3, for a sub-determination $\zeta_t = 011_2$, which drops the time aspect, and considering $\rho_H = 0.5$ so that $\mathbf{J}_{\rho}(T) = 0.5 \, \mathbf{J}(T)$, we have that

$$\mathbf{M}_{\zeta_t}(T) \left(\sum_{i=0}^{\infty} \mathbf{J}_{\rho}(T)^i \right) \mathbf{M}_{\zeta_t}(T)^T = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7.8 & 2.2 & 1.3 & 5.2 & 0 \\ 0 & 2.2 & 4.6 & 0.2 & 1.3 & 0 \\ 0 & 1.3 & 0.2 & 4.6 & 2.2 & 0 \\ 0 & 5.2 & 1.3 & 2.2 & 7.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}. \tag{66}$$

Algorithm 12 shows a combinational version of the sub-determined BFS. This procedure ensures that only paths present on the original MAG are considered on the sub-determined BFS.

The sub-determination of the results obtained from the BFS is done in the internal **if**, comprising lines 21 to 25 of Algorithm 12.

Algorithm 12: Sub-determined BFS.

```
   input  : J(H), τ(H), ζ and s ∈ 𝕍_ζ(H)
   output : vertices, distance, pred
 1 BFS-Sub(J(H), τ(H), ζ, s)
 2     n ← |𝕍(H)|  nS ← |𝕍_ζ(H)|
 3     T_ζ ← τ_ζ(H)                                   // ζ sub-determined companion tuple
 4     vertices ← vector of nS integers, all 0
 5     distance ← vector of nS integers, all ∞
 6     pred ← vector of nS integers, all Nil
 7     colorS ← vector of nS integers, all 0          // set all sub-determined vertices to white
       (unvisited)
 8     color ← vector of n integers, all 0            // set all vertices to white (unvisited)
 9     Q ← empty queue
10     for every v ∈ 𝕍(H) where D(v, τ_ζ(H)) = D(s, τ_ζ(H)) do
11         Enqueue(Q, D(v, τ(H)))
12         color[D(v, τ(H))] ← 1
13     end
14     vertices[D(s, T_ζ) − 1] ← 1
15     distance[D(s, T_ζ)] ← 0
16     while Q not empty do
17         u ← head[Q]
18         for each v successor of u do
19             if color[v] = 0 then
20                 color[v] ← 1                       // set vertex v to gray (visited)
21                 Enqueue(Q, v)
22                 if colorS[D(v, T_ζ)] = 0 then
23                     colorS[D(v, T_ζ)] ← 1          // set sub-determined vertex D(v, T_ζ) to gray
                       (visited)
24                     vertices[D(v, T_ζ)] ← 1
25                     distance[D(v, T_ζ)] ← distance[D(u, T_ζ)] + 1
26                     pred[D(v, T_ζ)] ← D(u, T_ζ)
27                 end
28             end
29         end
30         color[u] ← 2                               // set vertex u to black (closed)
31         Dequeue(Q)
32     end
33 return vertices, distance, pred
```

After applying Algorithm 12 to the MAG R with initial vertex $s = 1$ and sub-detemination $\zeta_R = 01_2$, the obtained result is

$$\begin{aligned} vertices &= [1, 2] \\ distances &= [0, 1, \infty] \\ pred &= [Nil, 1, Nil], \end{aligned} \tag{67}$$

which is consistent with the result obtained by Equation (63).

Further, applying Algorithm 12 to MAG T, shown in Figure 3, with starting composite vertex $s = (2, Bus)$ and applying the sub-determination $\zeta_t = 011_2$, which drops the time aspect, the obtained result is

$$\begin{aligned} vertices &= [2, 5, 3, 4] \\ distances &= [\infty, 0, 1, 2, 1, \infty] \\ pred &= [Nil, Nil, 2, 5, 2, Nil]. \end{aligned} \tag{68}$$

Considering that $\tau_{\zeta_t} = (3, 2)$ and $2 = D((2, Bus), (3, 2)), 5 = D((2, Subway), (3, 2))$, $3 = D((3, Bus), (3, 2))$, and $1 = D((1, Subway), (3, 2))$, this means that disregarding time, starting from $(2, Bus)$ it is possible to reach $(2, Subway)$ in 1 step, $(1, Subway)$ in 2 steps, and $(3, Bus)$ in 1 step. It is not possible to reach $(1, Bus)$ because there is no bus stop at location 1, neither $(3, Subway)$ because there is no subway station at location 3. From the predecessor list ($pred$) it is possible to build a BFS tree, where $(2, Bus)$ is the root, $(2, Subway)$ and $(3, Bus)$ are children of $(2, Bus)$, and $(1, Subway)$ is a child of $(2, Subway)$. Note that $(1, Subway)$ and $(3, Bus)$ are leaves. It can be seen that the result obtained in Equation (66) is consistent with the results obtained by Algorithm 12. Comparing Algorithm 12 to Algorithm 11, it can be seen that the main difference is the additional **for** loop at line 12 of Algorithm 12. Since the time complexity of this loop is $O(|\mathbb{V}(H)|)$, we then conclude that the time complexity of Algorithm 12 is $O(|\mathbb{V}(H)| + |E(H)|)$.

4.5. Depth-First Search (DFS)

In this section, we show the adaptation of the Depth-First Search (DFS) algorithm for use with MAGs. The DFS algorithm exposes many properties of the MAG structure and can be used as a primitive for the construction of many other algorithms [5]. We present DFS algorithms for both the full composite vertices representation of the MAG as well as for the sub-determined form. We remark that in the sub-determined algorithm the full information of the MAG is used, in the sense of preventing the use of paths that may exist in the sub-determined form of the MAG, while not actually existing in the original MAG.

4.5.1. DFS for Composite Vertices

The composite vertices implementation is constructed using the MAG's adjacency matrix $\mathbf{J}(H)$ and companion tuple $\tau(H)$. The implementation shown is very similar to the traditional implementation presented in [5], which is expected since the composite vertices representation of the MAG is indeed a directed graph, so that the original algorithm applies.

The proposed implementation can be seen in Algorithm 13 is similar to the original implementation. Therefore, considering the analysis provided in [5], we conclude that the time complexity of Algorithm 13 is $O(|\mathbb{V}(H)| + |E(H)|)$.

When applied to MAG T, shown in Figure 3, the DFS algorithm generates the result

$$\begin{aligned} d &= [0, 2, 22, 24, 3, 26, 28, 13, 19, 4, 12, 30, 32, 8, 14, 5, 7, 34] \\ f &= [1, 21, 23, 25, 18, 27, 29, 16, 20, 11, 17, 31, 33, 9, 15, 6, 10, 35] \\ pred &= [Nil, Nil, Nil, Nil, 2, Nil, Nil, 11, 2, 5, 5, Nil, Nil, 17, 8, 10, 10, Nil], \end{aligned} \tag{69}$$

where the list d carries the discovery time of each composite vertex, the list f the respective finish time of each composite vertex, and $pred$ the predecessor list of each composite vertex.

4.5.2. Sub-Determined DFS

The sub-determined DFS algorithm is presented in Algorithm 14 and is similar to the non sub-determined one. The main differences are at the Procedure Visit-DFS-Sub and the call to a sub-determined BFS at line 15 of the DFS-Sub function. This version for a sub-determined BFS is considered to determine the reachability of sub-determined vertices from the root of each

sub-determined DFS tree. This is necessary to prevent including vertices not reachable from the tree root in the non sub-determined MAG into the DFS trees constructed by Procedure Visit-DFS-Sub. An example of this is provided in Equation (71). The difference in Procedure Visit-DFS-Sub is that in addition to the root vertex for the DFS tree it also receives the reachability vector produced by the BFS. This reachability vector has one entry for each sub-determined vertex. This entry has value 1 when corresponding to a reachable vertex, while entries corresponding to unreachable vertices carry value 0.

Algorithm 13: DFS for composite vertices.

```
   input  : J(H), τ(H)
   output : discTime, finTime, pred
1  DFS(J(H), τ(H))
2      n ← |𝕍(H)|
3      for u = 1 to n do
4          color[u] ← 0                    // set all vertices to white (unvisited)
5          discTime[u] ← −1                // set discovery times to nil
6          finTime[u] ← −1                 // set finish times to nil
7          pred[u] ← −1                    // set predecessors to nil
8      end
9      time ← 0
10     for u = 1 to n do
11         if color[u] = 0 then
12             DFS-Visit(u)
13         end
14     end
15 return discTime, finTime, pred
1  Procedure DFS-Visit(u)
2      color[u] ← 1                        // set vertex u to gray (visited)
3      discTime[u] ← time
4      time ← time + 1
5      for each v successor of u do
6          if color[v] = 0 then
7              pred[v] ← u
8              DFS-Visit(v)
9          end
10     end
11     color[u] ← 2                        // set vertex u to black (closed)
12     finTime[u] ← time
13     time ← time + 1
```

To determine the time complexity of Algorithm 14, we consider that the sub-determined BFS executed at line 15 of Function DFS-Sub is done once for the root vertex of each sub-determined DFS tree. Since it is executed only once for each DFS tree, we conclude that the total time expended in the sub-determined BFS algorithm is $O(|\mathbb{V}(H)| + |E(H)|)$. Since the reachability check included in Function Visit-DFS-Sub is done by verifying the content of one entry of the reachability vector, it is done in $O(1)$ and therefore does not affect the overall time complexity of the Visit-DFS-Sub Function. Therefore, since the DFS is run upon the sub-determined MAG, it follows that the time complexity of doing the DFS part of the Algorithm is $O(|\mathbb{V}_\zeta(H)| + |E_\zeta(H)|)$. Since $|\mathbb{V}_\zeta(H)| < |\mathbb{V}(H)|$ and $|E_\zeta(H)| < |E(H)|$, we conclude that the time complexity is dominated by the BFS used for the reachability determination, making the overall time complexity of Algorithm 14 to be $O(|\mathbb{V}(H)| + |E(H)|)$.

Algorithm 14: Sub-determined DFS.

input : $\mathbf{J}(H)$, $\tau(H)$,ζ
output: *discTime*, *finTime*, *pred*

DFS-Sub($\mathbf{J}(H)$, $\tau(H)$,ζ)
  $T_\zeta \leftarrow \tau_\zeta(H)$ // ζ sub-determined companion tuple
  $\mathbf{M}_\zeta \leftarrow SubDetMatrix(H, \zeta)$
  $\mathbf{J}_\zeta = \mathbf{M}_\zeta\ \mathbf{J}(H)\ \mathbf{M}_\zeta^T$ // sub-determined adjacency matrix
  $n \leftarrow |\mathbb{V}_\zeta(H)|$ // number of sub-determined vertices
  for $u = 1$ *to* n **do**
    $color[u] \leftarrow 0$ // set all vertices to white (unvisited)
    $discTime[u] \leftarrow -1$ // set discovery times to nil
    $finTime[u] \leftarrow -1$ // set finish times to nil
    $pred[u] \leftarrow -1$ // set predecessors to nil
  end
  $time \leftarrow 0$
  for $u = 1$ *to* n **do**
    if $color[u] = 0$ **then**
      $vertices = BFS\text{-}Sub(\mathbf{J}(H), \tau(H), \zeta, T_\zeta)$
      $DFS\text{-}Visit\text{-}Sub(u, vertices)$
    end
  end
return *discTime*, *finTime*, *pred*

Procedure *DFS-Visit-Sub(u, vertices)*
  $color[u] \leftarrow 1$ // set vertex u to gray (visited)
  $discTime[u] \leftarrow time$
  $time \leftarrow time + 1$
  for *each v successor of u* **do**
    if $color[v] = 0$ **and** $vertices[v] \neq 0$ **then**
      $pred[v] \leftarrow u$
      $DFS\text{-}Visit\text{-}Sub(v, vertices)$
    end
  end
  $color[u] \leftarrow 2$ // set vertex u to black (closed)
  $finTime[u] \leftarrow time$
  $time \leftarrow time + 1$

When applying the sub-determined DFS algorithm to the example MAG T shown in Figure 3 with a sub-determination $\zeta_t = 011_2$, which drops the time aspect, the obtained result is

$$\begin{aligned} d &= [0, 2, 3, 6, 5, 10] \\ f &= [1, 9, 4, 7, 8, 11] \\ pred &= [Nil, Nil, 2, 5, 2, Nil], \end{aligned} \tag{70}$$

where the list d carries the discovery time of each sub-determined composite vertex, the list f its finish time and *pred* its predecessor.

Considering the MAG R shown in Figure 6a with a sub-determination $\zeta_R = 01_2$, the result obtained by Algorithm 14 is

$$\begin{aligned} d &= [0, 1, 4] \\ f &= [3, 2, 5] \\ pred &= [Nil, 1, Nil]. \end{aligned} \tag{71}$$

It can be seen that even though in the MAG R sub-determined by $\zeta_R = 01_2$ (see Figure 6b) there is a path from vertex 1 to 3, vertex 3 is not in the same DFS tree as vertices 1 and 2, even with the DFS starting at vertex 1, as can be seen in $d[0]$. This occurs because in MAG R (with no sub-determination) there is no path connecting the composite vertex 1 to the composite vertex 3.

5. Final Remarks

In this paper, we have presented the algebraic representation and basic algorithms of MultiAspect Graphs (MAGs). The key contribution has been to show that models based on the MAG abstraction (formally defined in [18]) can be represented by a matrix and a companion tuple. Furthermore, we have also shown that any possible MAG function (algorithm) can be obtained from this matrix-based representation. We have chosen to present the algebraic representation in matrix form since it is well accepted and known, thus contributing to the readability of the paper. Nevertheless, given the isomorphism property of MAGs, any representation form available for directed graphs can also be used for MAGs, including matrices as used here for convenience or sorted adjacency lists, for instance, that are a common representation of graphs in existing network analysis packages. This is an important theoretical result because it paves the way for adapting well-known graph algorithms for application in MAGs. In this sense, we have presented the adaptation for the MAG context of basic graph algorithms, such as computing degree, BFS, and DFS. These basic graph algorithms adapted to the MAG context can be used as primitives for building other more sophisticated MAG algorithms. Therefore, such examples can be seen as guidelines on how to properly derive MAG algorithms from basic algorithms on directed graphs. In particular, we have also presented the sub-determined versions of the same basic algorithms, showing that such versions disregard spurious paths that usually result from the sub-determination process, thus avoiding the pollution of the results with the consideration of such paths.

The discussed basic algorithms can be used as building blocks for other purposes, thus allowing the extension of the MAG applicability. Concerning algorithm complexity, given the isomorphism between a MAG and a traditional directed graph, we expect MAG algorithms to have the same complexity as the equivalent algorithms for traditional directed graphs. For instance, if the MAG algorithm is derived from a given traditional algorithm that is polynomial for traditional graphs, the resulting algorithm for MAGs will be polynomial in the size of the composite vertices representation of the MAG.

As future work, we intend to build upon the results here obtained for the algebraic representation and basic algorithms of MAGs to analyze MAG properties, such as the centrality of edges, composite vertices, and aspects. We also intend to consider the dynamics encountered in these properties in the cases where one of the MAG aspects represents time. Finally, we are also targeting the application of the MAG concept for the better understanding, modeling, and analysis of different real-world applications represented by high order complex networked systems.

Acknowledgments: This work was partially funded by the Brazilian funding agencies CAPES (STIC-AmSud Program), CNPq, FINEP, and FAPERJ as well as the Brazilian Ministry of Science, Technology, Innovations, and Communications (MCTIC).

Author Contributions: The problem definition, theoretical analysis, and algorithm design were performed jointly by all authors; Klaus Wehmuth derived the theorems and implemented the algorithms depicted in the paper. All authors read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Distel, R. *Graph Theory*, 4th ed.; Springer: Berlin/Heidelberg, Germany, 2010.
2. Jansson, J. Special Issue on Graph Algorithms. *Algorithms* **2013**, *6*, 457–458.
3. Deo, N. *Graph Theory with Applications to Engineering and Computer Science*, 1st ed.; Dover Publications: Mineola, NY, USA, 2016.

4. Tarjan, R. Depth-First Search and Linear Graph Algorithms. *SIAM J. Comput.* **1972**, *1*, 146–160.
5. Cormen, T.H.; Stein, C.; Rivest, R.L.; Leiserson, C.E. *Introduction to Algorithms*, 3rd ed.; MIT Press: Cambridge, MA, USA, 2009.
6. Friedkin, N.E. Theoretical foundations for centrality measures. *Am. J. Sociol.* **1991**, *96*, 1478–1504.
7. Wehmuth, K.; Ziviani, A. Distributed location of the critical nodes to network robustness based on spectral analysis. In Proceedings of the IEEE Latin American Network Operations and Management Symposium (LANOMS), João Pessoa, Brazil, 1–3 October 2011; pp. 1–8.
8. Takes, F.W.; Kosters, W.A. Computing the Eccentricity Distribution of Large Graphs. *Algorithms* **2013**, *6*, 100–118.
9. Wehmuth, K.; Ziviani, A. DACCER: Distributed Assessment of the Closeness CEntrality Ranking in complex networks. *Comput. Netw.* **2013**, *57*, 2536–2548.
10. Watts, D.; Strogatz, S.H. Collective dynamics of small-world networks. *Nature* **1998**, *393*, 440–442.
11. Barabási, A.L.; Albert, R. Emergence of Scaling in Random Networks. *Science* **1999**, *286*, 509–512.
12. Pastor-Satorras, R.; Vespignani, A. Epidemic Spreading in Scale-Free Networks. *Phys. Rev. Lett.* **2001**, *86*, 3200–3203.
13. Guimarães, A.; Vieira, A.B.; da Silva, A.P.C.; Ziviani, A. Fast Centrality-driven Diffusion in Dynamic Networks. In Proceedings of the 5th Annual Workshop on Simplifying Complex Networks for Practitioners, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 821–828.
14. Kurant, M.; Thiran, P. Layered Complex Networks. *Phys. Rev. Lett.* **2006**, *96*, doi:10.1103/PhysRevLett.96.138701.
15. Kivelä, M.; Arenas, A.; Barthelemy, M.; Gleeson, J.P.; Moreno, Y.; Porter, M.A. Multilayer networks. *J. Complex Netw.* **2014**, *2*, 203–271.
16. Leskovec, J.; Kleinberg, J.; Faloutsos, C. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In Proceedings of the 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 177–187.
17. Holme, P.; Saramäki, J. Temporal networks. *Phys. Rep.* **2012**, *519*, 97–125.
18. Wehmuth, K.; Fleury, E.; Ziviani, A. On MultiAspect graphs. *Theor. Comput. Sci.* **2016**, *651*, 50–61.
19. Scholtes, I.; Wider, N.; Garas, A. Higher-order aggregate networks in the analysis of temporal networks: Path structures and centralities. *Eur. Phys. J. B* **2016**, *89*, 1–15.
20. Benson, A.R.; Gleich, D.F.; Leskovec, J. Higher-order organization of complex networks. *Science* **2016**, *353*, 163–166.
21. Lucet, J.C.; Laouenan, C.; Chelius, G.; Veziris, N.; Lepelletier, D.; Friggeri, A.; Abiteboul, D.; Bouvet, E.; Mentre, F.; Fleury, E. Electronic Sensors for Assessing Interactions between Healthcare Workers and Patients under Airborne Precautions. *PLoS ONE* **2012**, *7*, doi:10.1371/journal.pone.0037893.
22. Xavier, F.H.Z.; Silveira, L.M.; Almeida, J.M.; Ziviani, A.; Malab, C.H.S.; Marques-Neto, H.T. Analyzing the Workload Dynamics of a Mobile Phone Network in Large Scale Events. In Proceedings of the First Workshop on Urban Networking (UrbaNe), Nice, France, 10–13 December 2012; pp. 37–42.
23. Blondel, V.D.; Decuyper, A.; Krings, G. A survey of results on mobile phone datasets analysis. *EPJ Data Sci.* **2015**, *4*, 1–55.
24. Karlebach, G.; Shamir, R. Modelling and analysis of gene regulatory networks. *Nat. Rev. Mol. Cell Biol.* **2008**, *9*, 770–780.
25. Yang, H.; Bell, M.G.; Meng, Q. Modeling the capacity and level of service of urban transportation networks. *Transp. Res. Part B Methodol.* **2000**, *34*, 255–275.
26. Bullmore, E.; Sporns, O. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nat. Rev. Neurosci.* **2009**, *10*, 186–198.
27. De Domenico, M.; Sasai, S.; Arenas, A. Mapping Multiplex Hubs in Human Functional Brain Networks. *Front. Neurosci.* **2016**, *10*, doi:10.3389/fnins.2016.00326.
28. Szell, M.; Lambiotte, R.; Thurner, S. Multirelational organization of large-scale social networks in an online world. *Proc. Natl. Acad. Sci. USA* **2010**, *107*, 13636–13641.
29. Wehmuth, K.; Ziviani, A.; Fleury, E. A unifying model for representing time-varying graphs. In Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, France, 19–21 October 2015; pp. 1–10.

30. Costa, E.C.; Vieira, A.B.; Wehmuth, K.; Ziviani, A.; da Silva, A.P.C. Time Centrality in Dynamic Complex Networks. *Adv. Complex Syst.* **2015**, *18*, doi:10.1142/S021952591550023X.
31. Sarraute, C.; Brea, J.; Burroni, J.; Wehmuth, K.; Ziviani, A.; Alvarez-Hamelin, J.I. Social Events in a Time-Varying Mobile Phone Graph. In Proceedings of the International Conference on the Scientific Analysis of Mobile Phone Datasets (NetMob), Cambridge, MA, USA, 8–10 April 2015.
32. Bang-Jensen, J.; Gutin, G.Z. *Digraphs: Theory, Algorithms and Applications*, 2nd ed.; Springer: London, UK, 2009.
33. Kepner, J.; Gilbert, J. *Graph Algorithms in the Language of Linear Algebra*; SIAM: Philadelphia, PA, USA, 2011.
34. Bapat, R.B. *Graphs and Matrices*, 2nd ed.; Springer: London, UK, 2014.
35. De Domenico, M.; Solé-Ribalta, A.; Cozzo, E.; Kivelä, M.; Moreno, Y.; Porter, M.; Gómez, S.; Arenas, A. Mathematical Formulation of Multilayer Networks. *Phys. Rev. X* **2013**, *3*, doi:10.1103/PhysRevX.3.041022.
36. Domenico, M.D.; Granell, C.; Porter, M.A.; Arenas, A. The physics of spreading processes in multilayer networks. *Nat. Phys.* **2016**, *12*, 901–906.