Formation reorganization by primitive operations on directed graphs.

Julien M. Hendrickx, Barış Fidan, Changbin Yu, Brian D.O. Anderson and Vincent D. Blondel

***Abstract*— In this paper, we study the construction and transformation of two-dimensional persistent graphs. Persistence is a generalization to directed graphs of the undirected notion of rigidity. Both notions are currently being used in various studies on coordination and control of autonomous multi-agent formations. In the context of mobile autonomous agent formations, persistence characterizes the efficacy of a directed formation structure with unilateral distance constraints seeking to preserve the shape of the formation. Analogously to the powerful results about Henneberg sequences in minimal rigidity theory, we propose different types of directed graph operations allowing one to sequentially build any minimally persistent graph (i.e. persistent graph with a minimal number of edges for a given number of vertices), each intermediate graph being also minimally persistent. We also consider the more generic problem of obtaining one minimally persistent graph from another, which corresponds to the on-line reorganization of the sensing and control architecture of an autonomous agent formation. We prove that we can obtain any minimally persistent formation from any other one by a sequence of elementary local operations such that minimal persistence is preserved throughout the reorganization process. Finally, we briefly explore how such transformations can be performed in a decentralized way.**

I. Introduction

The recent progress in the field of autonomous agent systems has led to new problems in control theory [2], [4], [19] and graph theory [6], [11], [17]. By autonomous agent, we mean here any human controlled or unmanned vehicle that can move by itself and has a local intelligence or computing capacity, such as ground robots, air vehicles or underwater vehicles. The results derived in this paper concern autonomous agents evolving in a two dimensional space.

J. M. Hendrickx (corresponding author) and V. D. Blondel are with Department of Mathematical Engineering, Université catholique de Louvain, Avenue Georges Lemaitre 4, B-1348 Louvain-la-Neuve, Belgium; `hendrickx,blondel@inma.ucl.ac.be`. Their work is supported by the Belgian Programme on Interuniversity Attraction Poles initiated by the Belgian Federal Science Policy Office, and The Concerted Research Action (ARC) “Large Graphs and Networks” of the French Community of Belgium. The scientific responsibility rests with its authors. Julien Hendrickx holds a FNRS fellowship (Belgian Fund for Scientific Research). This work was supported in part by DoD AFOSR URI for ”Architectures for Secure and Robust Distributed Infrastructures,” F49620-01-1-0365 (led by Stanford University).

B. Fidan, C. Yu and B. Anderson are with Australian National University and National ICT Australia , 216 Northbourne Ave, Canberra ACT 2601 Australia ; `baris.fidan,brad.yu,brian.anderson@nicta.com.au`. Their work is supported by an Australian Research Council Discovery Project Grant and by National ICT Australia, which is funded by the Australian Government’s Department of Communications, Information Technology and the Arts and the Australian Research Council through the Backing Australia’s Ability Initiative. Changbin Yu was an Australia-Asia Scholar supported by the Australian Government’s Department of Education, Science and Training through Endeavours Programs.

A. Formations and rigid graphs

Many applications require the shape of a multi-agent formation to be preserved For example, target localization by a group of unmanned airborne vehicles (UAVs) using either angle of arrival data or time difference of arrival information appears to be best achieved (in the sense of minimizing localization error) when the UAVs are located at the vertices of a regular polygon [5]. Other examples of optimal placements for groups of moving sensors can be found in [16]. This objective can be achieved by explicitly keeping *some* inter-agent distances constant. In other words, some inter-agent distances are explicitly maintained constant so that all the inter-agent distances remain constant. The information structure arising from such a system can be efficiently modelled by a graph, where agents are abstracted by vertices and actively constrained inter-agent distances by edges.

Such a graph is said to be *rigid* if the corresponding set of distance constraints is sufficient to maintain the formation shape. In other words, a graph is rigid if provided that all prescribed distance constraints are satisfied during a continuous displacement, all inter-agent distances remain constant, as shown in Figure 1. This property depends indeed almost only on the graph of distance constraints, and not on the particular agents positions and inter-agents distance (see [20] for more details on this subject). Note that this notion of rigidity also represents the rigidity of a framework where the vertices correspond to joints and the edges to bars.

B. Formations with unilateral distance constraints

Unlike in the case of frameworks where distance constraints are guaranteed by the presence of bars between joints, constraints on inter-agent distances in formations have to be maintained by means of measurements and control actions. A distance between two agents can be cooperatively maintained by the two agents, in which case the rigidity theory can directly be applied. But one can also give the full responsibility of maintaining the constraint to one agent, which has to maintain its distance from the other constant, this latter agent being unaware of that fact and taking therefore no specific action helping to satisfy the distance constraint. This unilateral character can be a consequence of the technological limitations of the autonomous agents. Some UAV’s can for example not efficiently sense objects that are behind them or have an angular sensing range smaller than 360° [3], [8], [18]. Also, some of the authors of this paper are working with agents in which optical sensors have blind three dimensional cones. It can also be desired to ease the trajectory control of the formation, as it allows so-called leader follower formations [2], [7], [19]. In such a formation, one agent (leader) is free of inter-agent constraints and is only constrained by the desired trajectory of the formation, and a second agent (first follower) is responsible for only one distance constraint and can set the relative orientation of the formation. The other agents have no decision power and are forced by their distance constraints to follow the two first agents. An example of such a formation is shown in Figure 2. Finally, it has been argued [2] that for some classes of control law, having the distance constraints maintained by both agents can lead to unstable behaviors in the presence of measurement errors (It is however possible to avoid such behavior by introducing dead-zones at the cost of limited inaccuracy in the preservation of formation shape [9]).

A structure of unilateral distance constraints can be represented using a directed graph, a vertex being connected to another vertex by a directed edge if the agent corresponding to the first vertex has to maintain its distance from the agent represented by the second vertex. The characterization of the directed information structures which can efficiently maintain the formation shape has begun to be studied under the name of “directed rigidity” or “rigidity of a directed graph” [1], [2], [6]. These works included several conjectures about minimal directed rigidity, i.e., directed rigidity with a minimal number of edges for a fixed number of vertices. In [11], Hendrickx et al. proposed a theoretical framework to analyze these issues, where the name of “persistence” was used in preference to “directed rigidity”, since the latter notion does not correspond to the immediate transposition of the undirected notion of rigidity to directed graphs. The intuitive definition of persistence is the following: An information structure is persistent if, provided that each agent is trying to satisfy all the distance constraints for which it is responsible, all the inter-agent distances remain constant and as a result the formation shape is preserved. It is shown in [11] that persistence is actually the conjunction of two distinct notions: rigidity of the underlying undirected graph (i.e. the graph obtained by ignoring the direction of the edges), and constraint consistence. Constraint consistence of an information structure means that, provided that each agent is trying to satisfy all its distances constraints, all the agents actually succeed in doing so. In other words, no agent has an impossible task, as shown in the example in Figure 3. Observe that this last notion strongly depends on the directed structure of the graph, while rigidity only relies on its underlying undirected graph. An example of a persistent graph is provided in Figure 2. For agents evolving in a two-dimensional space, a purely combinatorial criterion to decide persistence is provided in [11].

C. Building formations with minimally persistent graphs

In this paper, we focus on minimally persistent graphs, i.e. persistent graphs having a minimal number of edges for a given number of vertices, and their connections with minimally rigid graphs. More particularly *we analyze different ways to sequentially build minimally persistent graphs, analogously to the Henneberg sequences for the minimally rigid graphs* [14], [20]. It has indeed long been known that every minimally rigid graph can be obtained from the complete graph on two vertices by a sequence of two basic operations, as detailed in Section II. The natural extension of these operations to directed graphs [7] does *not* allow one to build all minimally persistent graphs, as remarked in [11] and reviewed in Section IV-B. For reasons reviewed in Section III-B it is however desirable to have a set of operations for the building of *all* minimally persistent graphs. Such a set is indeed needed to develop an efficient way to cope with the loss of one or several agents, as it would allow adding or removing an agent in a formation in a decentralized way. A second quite different motivation is that in the presence of measurement errors, reorganization of a formation may also be needed to cope with some ill-conditioned system without modifying the relative positions of the agents. Both issues are especially relevant if one considers that a formation needs to evolve dynamically with the external conditions, modifying for example its shape and or its leadership structure.

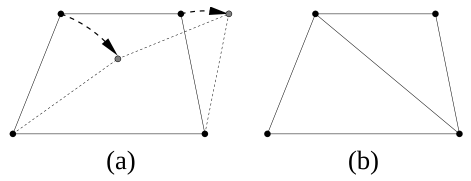

Fig. 1. Representation of (a) a non-rigid and (b) a rigid graph/formation. The solid structure in (a) can indeed be deformed to the dotted structure without breaking any distance constraint.

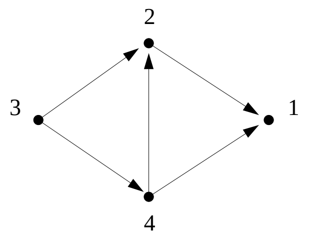

Fig. 2. Representation of a persistent graph, i.e., a rigid constraint consistent graph. This graph corresponds also to a leader-follower formation, where 1 is the leader and 2 the first follower.

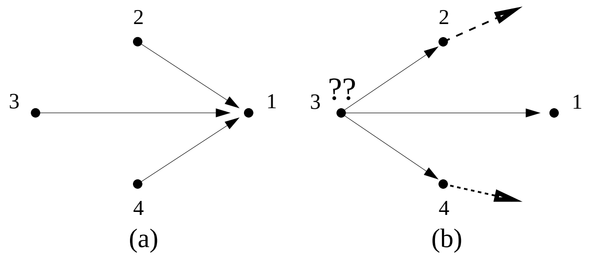

Fig. 3. Representation of (a) a constraint consistent and (b) a non-constraint consistent (in 2 dimension) graph/formation. One can indeed see in (b) that for almost any uncoordinated continuous displacement of the agents 2 and 4 (which are unconstrained), the agent 3 is unable to move in such a way that it maintains its distances to all of 1,2 and 4 constant. However, such a situation could not happen in graph (a).

We prove some characteristics of the operation sets allowing one to build all minimally persistent graphs, and provide one of the simplest sets achieving this goal (Another set, for which the number of operations required to build a minimally persistent graph is uniquely determined by the number of its vertices, can be found in [12], [13]). We also consider the more generic problem of obtaining one persistent graph from another. From an autonomous agent point of view, this corresponds to an on-line reorganization of the agent formation. Note that although the notion of persistence has been also defined in three or higher dimensions [22], the present analysis only concerns two-dimensional persistence, i.e., the persistence of graphs representing the information structure of a formation evolving in a two-dimensional space. Extension to the three dimensional case may be difficult; even for undirected graphs, three-dimensional Henneberg sequences theory is indeed incomplete.

D. Outline of the paper

In Section II, we review the main properties of the Henneberg sequences for minimally rigid graphs and of the two operations - vertex addition and edge splitting - on which it is based. Section III briefly reviews minimal persistence, and details the different reasons for which a directed version of Henneberg sequences is desirable. We consider in Section IV the natural extension of the vertex addition and edge splitting to directed graphs, and show that although they preserve minimal persistence, they are not sufficient to build all minimally persistent graphs, and do therefore not constitute a complete generalization of Henneberg sequences to directed graphs. We also show that any set of directed operations based on the undirected vertex addition and edge splitting operations and allowing one to build all minimally persistent graphs *must* contain non-confined operations. Non-confined operations are operations reversing the directions of edges that are not affected by the corresponding operation for undirected graphs. In other words operations that, in addition to adding or removing vertices and edges, reverse the directions of one or more edges. This analysis is done by reasoning on reverse construction of persistent graphs using reverse operations. In Section V we introduce the simplest non-confined operation, edge reversal, and show how it can be used to reach the goal of building all minimally persistent graphs. We see that, unlike when building minimally rigid undirected graphs with Henneberg sequences, the required number of operations is not uniquely determined by the size of the graph. We also explore the possibility of performing some of the operations in a decentralized way, something which is of critical importance from an application point of view. Finally, this paper ends with the concluding remarks of Section VI. Note that a more detailed explanation of the undirected and directed versions of the Henneberg operations can be found in [12], [13], together with an alternative set of four operations allowing one to obtain all minimally persistent graphs.

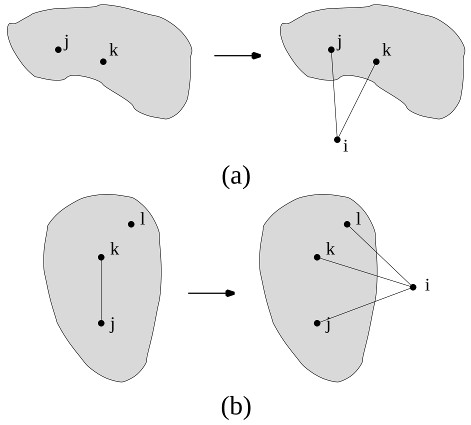

Fig. 4. Representation of (a) undirected vertex addition operation and (b) edge splitting operation.

II. Minimally rigid graphs and undirected Henneberg sequences

In this section, all graphs are considered as undirected, but in the rest of this paper, they are always assumed to be directed. Although all the definitions and results of this section are given for undirected graphs, they can also be applied to directed graphs. If G is a directed graph, we call the *underlying undirected graph* of G the undirected graph obtained by ignoring the directions of the edges of G.

The intuitive meaning of the undirected notion of rigidity is explained in the Introduction. For a more formal definition, the reader is referred to [11], [20]. In $\Re^2$, there exists a combinatorial criterion to check if a given graph is rigid (Laman's theorem [15], [21]). A *minimally rigid* graph is a rigid graph such that no edge can be removed without losing rigidity. A key intermediate result in Laman's Theorem proof [15] is the following criterion:

Proposition 1: A graph $G = (V, E)$ ($|V| > 1$) is minimally rigid if and only if $|E| = 2|V| - 3$ and for all $E'' \subseteq E, E'' \neq \varnothing$, there holds $|E''| \leq 2|V(E'')| - 3$, where $V(E'')$ is the set of vertices incident to E''.

Let j, k be two distinct vertices of a minimally rigid graph $G = (V, E)$. A *vertex addition* operation consists in adding a vertex i, and connecting it to j and k, as shown in Figure 4(a). It follows from Proposition 1 that this operation preserves minimal rigidity. Moreover, if a vertex i has degree 2 in a minimally rigid graph, one can always perform the inverse vertex addition operation by removing it (and its incident edges) and obtain a smaller minimally rigid graph.

Let j, k, l be three vertices of a minimally rigid graph such that there is an edge between j and k. An *edge splitting* operation consists in removing this edge, adding a vertex i and connecting it to j, k and l, as shown in Figure 4(b). This operation provably preserves minimal rigidity [20]. Consider now a vertex i connected to three vertices j, k and l. A reverse edge splitting consists in removing i and adding one edge among (j, k), (k, l) and (l, j), *in such a way that the*

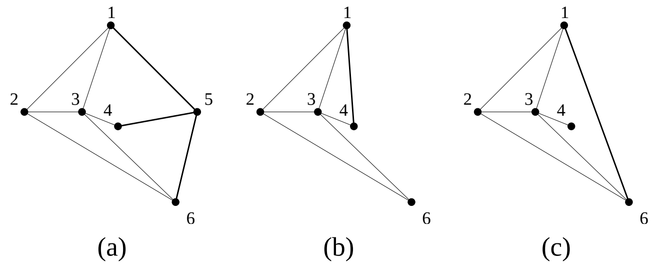

Fig. 5. Example of unfortunate added edge selection in reverse edge splitting. After the removal of the vertex 5 from the minimally rigid graph (a), minimal rigidity can be preserved by the addition of the edge $(1,4)$ but not of $(1,6)$, as shown respectively on (b) and (c). In the latter case, the subgraph induced by 1, 2, 3 and 6 contains indeed 6 edges and 4 vertices ($6 > 2.4 - 3 = 5$), and the edge $(3,4)$ is only fixed to the graph by one of its vertices.

graph obtained is minimally rigid. This operation can be performed on every vertex with degree 3 in a minimally rigid graph [15], [20], but one cannot freely choose the edge to be added as shown on the example in Figure 5.

A *Henneberg sequence* is a sequence of graphs $G_2, G_3, \ldots, G_{|V|}$ with $G_2 = K_2$ being the complete graph on two vertices and where each graph G_i ($i \geq 3$) can be obtained from G_{i-1} by either a vertex addition operation or an edge splitting operation. Since these operations preserve minimal rigidity and since K_2 is minimally rigid, every graph in such a sequence is minimally rigid.

Theorem 1: [20] Every minimally rigid graph on more than one vertex can be obtained as the result of a Henneberg sequence. Moreover, all intermediate graphs of such a sequence are minimally rigid.

III. Minimally persistent graphs

A. Review of minimal persistence

Consider a group of autonomous agents represented by vertices of a graph. To each of these agents, one assigns a (possibly empty) set of unilateral distance constraints represented by directed edges: the notation (i,j) for a directed edge connotes that the agent i has to maintain its distance to j constant during any continuous move. The persistence of the directed graph means that provided that each agent is trying to satisfy its constraints, the distance between any pair of connected or non-connected agents is maintained constant during any continuous move, and as a consequence the shape of the formation is preserved. A formal definition of persistence is given in [11].

A graph is *minimally persistent* if it is persistent and if no edge can be removed without losing persistence. The following result provides a swift criterion to decide minimal persistence:

Proposition 2: [11] A graph is minimally persistent if and only if it is minimally rigid and no vertex has an out-degree larger than 2.

We call the *number of degrees of freedom* of a vertex i the (generic) dimension of the set in which the corresponding agent can choose its position when all the others positions are fixed. It thus represents in some sense the decision power of this agent. In a two-dimensional space, an agent having two or more distance constraints to satisfy (out-degree 2) has only up to two possible positions. It has therefore no degree of freedom. An agent having only one distance constraint to satisfy (out-degree 1) can move on a circle centered on its neighbor, and has thus one degree of freedom. Finally an agent having no distance constraint to satisfy (out-degree 0) can move freely in the plane and has therefore two degrees of freedom. The number of degrees of freedom of a vertex i in a directed graph is thus given by $\max(0, 2 - d^+(i))$ (where $d^+(i)$ and $d^-(i)$ represent respectively the out- and in-degree of the vertex i). As a consequence of Proposition 2, the number of degrees of freedom of a vertex i in a minimally persistent graph is $2 - d^+(i)$. The total number of degree of freedom in a minimally persistent graph $G(V,E)$ is $\sum_{i\in V}(2 - d^+(i)) = 2|V| - \sum_{i\in V} d^+(i) = 2|V| - |E|$. It follows then from Propositions 1 and 2 that this number is always 3 in a minimally persistent graph. This result is consistent with the intuition, there are indeed three degrees of freedom to choose the position and orientation of a rigid body in a 2-dimensional space.

B. Applications-type motivations for directed versions of Henneberg sequences

Having a set of operations allowing one to sequentially build all minimally persistent graphs in a systematic way analogously to Henneberg sequences for undirected graphs, or to reorganize any minimally persistent graph into any other is an interesting result from a theoretical point of view. But it also has several practical implications, which we review in this section.

Such a set, if simple enough, could first provide a decentralized way to add an agent to a formation. This would for example be relevant in a situation where a few additional agents are needed to help a formation to cope with unplanned task. The dual problem of an agent leaving the formation is equally relevant. An agent may indeed need to leave the formation once it has accomplished its task within the formation or to fulfill a particular temporary mission out of the formation. Also, in a large formation the possibility of losing an agent cannot be excluded, due to technical malfunctions or to an hostile action for example. This generally leads to a loss of persistence, and an efficient method is thus needed to reconfigure the formation in order to recover persistence. This problem is known as the *closing ranks problem* and happens to be a particular case of the *splitting problem* in which a formation is split in two or more subsets, each of them potentially needing to reconfigure its distance constraints in order to be persistent. These issues are addressed in the undirected case by Eren et al. [6], and the proposed solution relies on the undirected Henneberg sequences (Actually, a modest extension of the underlying theory is needed). Therefore it is reasonable to suppose that a directed analogous to Henneberg sequences would be helpful

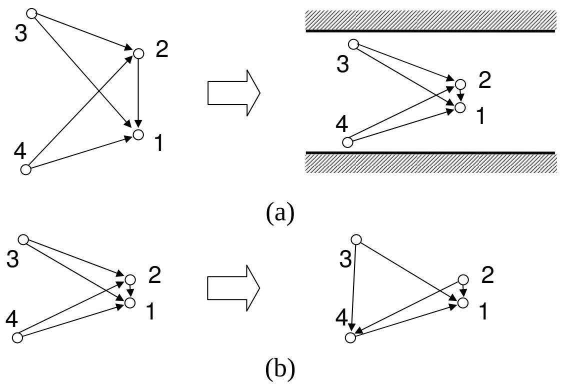

Fig. 6. (a) Formation needing to reduce the width of its shape in order to go through a narrow passage. As a result, the distance constraints sets of agents 3 and 4 become ill-conditioned. This can be solved by reorganizing the structure of constraints without modifying the formation shape (b).

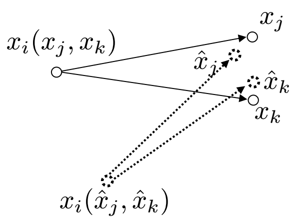

Fig. 7. Representation of an agent with an ill-conditioned set of constraints. Small positions variations or measurement errors $\hat{x}_j - x_j$, $\hat{x}_k - x_k$ can cause large modifications of the desired position of i.

in solving the directed version of the closing ranks problem.

The goal of persistence is to maintain the shape of a formation during its displacement. This must however not hide the possible need for this shape to be modified due to a varying external environment. Suppose for example that a formation has to traverse a narrow passage to avoid a dangerous zone such as mountains or a fire, or to avoid the detection range of some radars. The formation width needs to be reduced as represented in Figure 6(a), but it might be desirable to conserve its length. Such a shape modification can lead to instabilities inside the formations. An agent i having distance constraints toward two agents j and k has theoretically a position uniquely determined (up to an axial symmetry) by the positions of j and k. But in a real and noisy environment, determining the position of i can become an ill-conditioned problem if the angle $j\hat{i}k$ becomes too small as represented in Figure 7. Due to a shape modification an initially sound formation can become ill-conditioned as in Figure 6(a). This problem could be prevented by imposing some stability ensuring conditions on the new agent relative positions. However, in several cases, a simple reorganization of the distance constraint structure can re-improve the conditioning of the formation systems without modifying the relative positions of the agents as presented in Figure 6(b).

Finally, a shape modification or variation in the external environment can also lead to the necessity of modifying the formation leadership without necessarily modifying the undirected structure of the distance constraints. This could

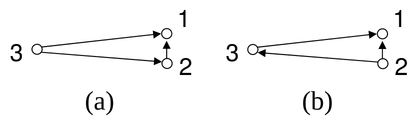

Fig. 8. Representation of two minimally persistent formations having the same undirected constraints. However (b) is much better conditioned that (a) has the smallest angle between two constraints for which the same agent is responsible is much larger than in (a).

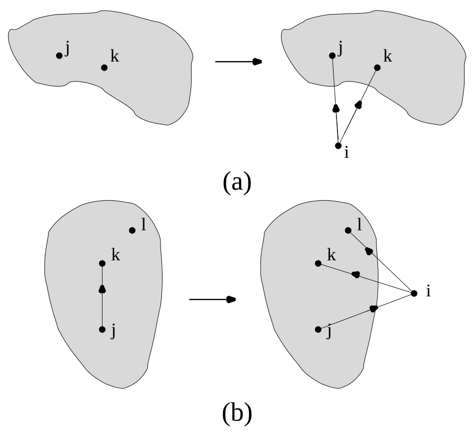

Fig. 9. Representation of the directed vertex addition (a) and edge splitting (b).

be the case for simple control reasons as in the example represented in Figure 8. Or, it might happen that in the course of the formation movement, some agents get an easier or more accurate access to information that could or should influence the formation's desired path. Without necessarily influencing the undirected structure of the formation, it would then be beneficial to provide this agent with some decision power or degree of freedom, that is to have the agent actively maintain less than two constraints.

IV. Natural extension of the Henneberg operations

A. Definition of the operations

Let j, k be two distinct vertices of a minimally persistent graph $G = (V, E)$. A *directed vertex addition* [7], [10] consists in adding a vertex i and two directed edges (i, j) and (i, k) as shown in Figure 9(a). A reverse (directed) vertex addition consist in removing a vertex with an out-degree 2 and an in-degree 0 from a minimally persistent graph.

Lemma 1: The directed vertex addition and reverse directed vertex addition operations preserve minimal persistence.

Proof: From an undirected point of view, both operations preserve minimal rigidity. Moreover, the directed vertex addition operation adds a vertex with out-degree 2 without affecting other vertex out-degrees, and the reverse directed vertex addition operation removes a vertex without affecting other vertex out-degrees. It follows then from Proposition 2 that both operations preserve minimal persistence

■

Let (j, k) be a directed edge in a minimally persistent graph and l a distinct vertex. A *directed edge splitting* [7], [10] consists in adding a vertex i, an edge (i, l), and replacing the edge (j,k) by (j,i) and (i,k), as shown in Figure 9(b). Let now i be a vertex with out-degree 2 and in-degree 1, call j the vertex left by an edge arriving at i, and k,l the other neighbors of i. its neighbors j,k and l. The reverse directed edge splitting operation consists in removing i and its incident edges, and adding either (j,k) or (j,l) (k and l being interchangeable) *in such a way that the graph obtained is minimally rigid.*

Lemma 2: The directed edge splitting and reverse directed edge splitting operations preserve minimal persistence.

Proof: From an undirected point of view, the edge splitting operation preserves minimal rigidity. It follows from its definition that the reverse directed edge splitting operation also does. Again, these operations add or remove a vertex with out-degree 2 without affecting the other vertex out-degrees. So it follows from Proposition 2 that both operations preserve minimal persistence.

■

We denote by $\mathcal{S}$ the set of operations containing the directed vertex addition operation and the directed edge splitting operations and by $\mathcal{S}^{-1}$ the operation set containing their reverse versions (the same convention is used in the sequel for all operation sets). The smallest minimally persistent graph on more than one vertex consists in two vertices connected by one directed edge, its minimal persistence follows directly from Propositions 2 and 1. We refer to this graph as a *leader-follower pair*, the leader being the vertex with an out-degree 0. Since the operations in $\mathcal{S}$ preserve minimal persistence, any graph obtained by performing a sequence of directed vertex addition or edge splitting operations on an initial leader-follower pair is minimally persistent. The following result establishes that to any minimally rigid graph corresponds a minimally persistent graph that can be obtained in that way, as already argued in [11].

Proposition 3: It is possible to assign directions to the edges of any minimally rigid graph such that the obtained directed graph is minimally persistent and can be obtained by performing a sequence of operation in $\mathcal{S}$ on an initial leader-follower pair. Moreover, all intermediate graphs are minimally persistent.

Proof: Let G be a minimally rigid (undirected) graph. By Theorem 1, it can be obtained by performing a sequence of undirected vertex additions and edge splittings on K_2. By performing the same sequence of the directed version of these operations on an initial leader-follower pair, one obtains a directed graph having G as underlying undirected graph. Moreover, since this initial seed is minimally persistent and since the directed versions of both vertex addition and edge splitting preserve minimal persistence, the obtained graph and all the intermediate graphs are minimally persistent.

■

B. Insufficiency of the natural extension

We now show that the operations in $\mathcal{S}$ do not allow one to grow all minimally persistent graphs from an initial seed.

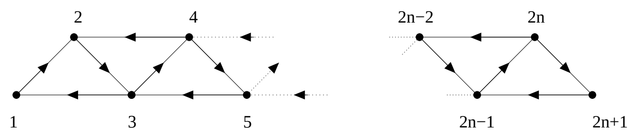

Fig. 10. Class of graphs on which no reverse vertex addition or edge splitting can be performed.

Consider the graph in Figure 10 for $n > 1$. This graph is minimally rigid as its undirected structure can be obtained from K_2 by performing $2n-1$ undirected vertex additions, connecting each new vertex i to $i-1$ and $i-2$. Moreover, no vertex has an out-degree larger than 2; by Proposition 2 it is thus minimally persistent. If this graph could be obtained by performing an operation in $\mathcal{S}$ on a smaller minimally persistent graph, then it would be possible to re-obtain this smaller graph by applying an operation in $\mathcal{S}^{-1}$. Observe that no vertex has an in-degree 0; it is thus impossible to perform a reverse vertex addition operation. Moreover, only the vertex $2n$ satisfies the required conditions about the in- and out-degree in order to offer the possibility of removal by a reverse edge splitting operation. Applying this reverse operation would consist in removing $2n$ and adding either the edge $(2n+1, 2n-1)$ or the edge $(2n-2, 2n-1)$. But the opposite edges $(2n-1, 2n+1)$ and $(2n-1, 2n-2)$ are already present in the graph, so that adding $(2n+1, 2n-1)$ or $(2n-1, 2n+1)$ would create a cycle of length 2. It follows from a direct application of Proposition 1 that a graph containing such a cycle is never minimally rigid and therefore never minimally persistent. Note that adding $(2n-2, 2n+1)$ or $(2n+1, 2n-2)$ would restore the graph rigidity, but the operation would then not be a reverse directed edge splitting such as defined above and would not be out-degree preserving for the vertices remaining in the graph, and would then not necessarily preserve minimal persistence. In the first case, the out-degree of $2n-2$ would indeed be increased to 3, preventing the graph obtained from being (minimally) persistent. The possibility of using such operations is further explored in Section IV-C. Furthermore since this reasoning holds for any $n > 1$, we have an infinite class of graphs on which none of the two reverse operations in $\mathcal{S}^{-1}$ can be performed. (A minimally persistent graph in which one vertex has two degrees of freedom and another one one degree of freedom and which cannot be obtained from a smaller minimally persistent graph by an operation in $\mathcal{S}$ can be found in [11]) As a consequence, it is not possible to build every minimally persistent graph by performing a sequence of operations in $\mathcal{S}$ on some seed graph taken in a finite set of graphs. Observe also that unlike in the case of undirected reverse operations for minimally rigid graphs, there are vertices in minimally persistent graphs that cannot be removed by a reverse (directed) edge splitting even though they satisfy the degree condition, i.e. they have an out-degree 2 and an in-degree 1.

C. Necessary involvement of external edges.

It is shown in the previous sections that the immediate generalizations to directed graphs of the undirected vertex addition and edge splitting operations are not sufficient to

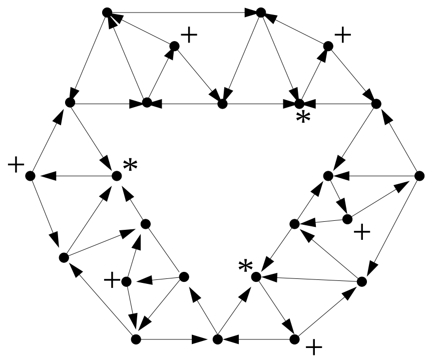

Fig. 11. A minimally persistent graph no vertex of which can be removed without losing persistence by a reverse (generalized) vertex addition or a confined (generalized) reverse edge splitting. The symbol "*" represents one degree of freedom. Vertices that are candidate to be removed by a reverse generalized edge splitting are labelled "+".

build all minimally persistent graphs. These operations are designed to preserve the out-degree of the already existing vertices, so that they can be performed (preserving minimal persistence) on any minimally persistent graph, regardless of the out-degree of the vertices to which the added vertex is connected. But one could imagine other generalizations of the undirected operations, which would for example increase some out-degree and therefore could only be applied under restricted conditions. For example one can contemplate adding a vertex i with in-degree and out-degree 1, and edges (i, j) and (k, i). As a result, the out-degree of k is increased by 1. This operation preserves minimal persistence if and only if k has a degree of freedom before the addition, that is if the out-degree of k is smaller than 2 before the addition. In the sequel, we adopt the terms *generalized vertex addition* and *generalized edge splitting* for any operation which is equivalent to a vertex addition or an edge splitting from an undirected point of view. An operation is said to be *confined* if it only affects edges that are involved in the corresponding undirected operation. In other words, an operation is confined if it only consists in addition or deletion of edges and vertices, and not in the reversion of edges directions, these reversions having indeed no undirected counterparts. For example, both operations in $\mathcal{S}$ are confined. We now prove that it is impossible to obtain all minimally persistent graph by applying a sequence of confined generalized vertex addition or edge splitting operations to an initial leader-follower seed.

Proposition 4: If a set exists of generalized vertex additions and edge splittings allowing one to build all minimally persistent graphs from an initial leader-follower seed, such a set must contain a non-confined edge splitting.

Proof: Suppose that one wants to remove a vertex without losing persistence from the provably minimally persistent graph represented in Figure 11 using a generalized reverse edge splitting or reverse vertex addition. The only ones that can be removed are those with (undirected) degree 2 or 3, and they are shown with a label "+". As they have undirected degree 3, a generalized reverse edge splitting operation would be needed. Suppose now that one wants to use a confined version of this operation. One would then remove one of the vertices with a label "+" and connect two of its neighbors by a directed edge. Observe that among the three pairs of neighbors of any vertex with a label "+", two are already connected, and the last pair contains two vertices with an out-degree 2. Adding an edge between a pair of neighbors of the removed vertex without reversing the direction of any other edge would thus imply the presence of either a vertex with out-degree 3 which by Theorem 2 is impossible in a minimally persistent graph, or of a cycle of length 2 which by Proposition 1 cannot appear in a minimally rigid graph. This removal should therefore be performed by a *non-confined* reverse generalized edge splitting. ■

Such a set of operations containing non-confined edge splitting and allowing one to build all minimally persistent graphs starting from a leader-follower seed can be found in [12] and [13]. Since one vertex is added at each operation, the number of operations required to obtain a graph $G = (V, E)$ is $|V| - 2$. The existence of confined operations that would not be equivalent to vertex addition or edge splitting, but that would however preserve minimal persistence and allow one to build all minimally persistent graphs with $|V|$ vertices in $|V| - 2$ operations starting with a leader-follower seed remains an open question. Such operations would have to be proved to preserve minimal rigidity.

V. A PURELY DIRECTED OPERATION

We have shown that unless we use operations which are not equivalent to Henneberg sequence operations from an undirected point of view, the use of non-confined operations is required to be able to build all minimally persistent graphs. We therefore now introduce the edge reversal operation, the simplest possible non-confined operation, which is neutral from an undirected point of view as it only reverses the direction of one edge. We then define two macro-operations which help us to prove two properties. First, edge reversal operations are sufficient to obtain any minimally persistent graph from any other one having the same underlying undirected graph. Second, edge reversal operations combined with those in $\mathcal{S}$ are sufficient to obtain any minimally persistent graph from a unique initial seed.

A. Edge reversal

Let (i, j) be an edge such that j has at least one degree of freedom, i.e., $d^+(j) = 0$ or $d^+(j) = 1$. The *edge reversal* operation consists in replacing the edge (i, j) by (j, i). As a consequence, one degree of freedom is transferred from j to i. This operation is its auto-inverse and preserves minimal persistence since it does not affect the underlying undirected graph and the only increased out-degree $d^+(j)$ remains no greater than 2. From an autonomous agent point of view j transfers its decision power or a part of it to i.

B. Path reversal

Given a directed path P between a vertex i and a vertex j such that j has a positive number of degrees of freedom,

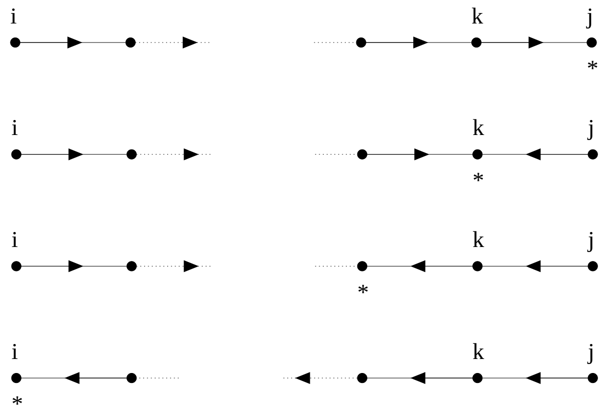

Fig. 12. Implementation of the path reversal by a sequence of edge reversals. The symbol "*" represents one degree of freedom.

a *path reversal* consists in reversing the directions of all the edges of P. As a result, j loses a degree of freedom, i acquires one, and there is a directed path from j to i. Moreover, the number of degrees of freedom of all the other vertices remain unchanged. Note that i and j can be the same vertex, in which case the path either has a trivial length 0 or is a cycle. In both of these situations, the number of degrees of freedom is preserved for every vertex.

The path reversal can easily be implemented with a sequence of edge reversals: Since j has a degree of freedom, one can reverse the last edge of the path, say (k, j), such that j loses one degree of freedom while k acquires one. One can then iterate this operation along the path until i, as shown in Figure 12. At the end, i has an additional degree of freedom, j has lost one, and all the edges of the paths have been reversed. Note that the sequence of edge reversals can usually not be performed in another order, for the condition requiring the availability of a degree of freedom would not be satisfied. The final result would be the same, but all the intermediate graphs would not necessarily be minimally persistent.

The following lemma, which is a particular case of a result available in [22], implies that a path reversal operation allows the transfer of a degree of freedom from any vertex having at least one degree of freedom to any other vertex having less than two of them.

Lemma 3: Let G be a minimally persistent graph, i and j two vertices of G with $d^+(i) \geq 1$ and $d^+(j) \leq 1$. Then, there is a directed path from i to j.

C. Cycle reversal

A *cycle reversal* consists in reversing all the edges of a directed cycle. This operation does not affect the number of degrees of freedom of any vertex nor the underlying undirected graph, and preserves therefore minimal persistence.

A cycle reversal on a minimally persistent graph can be implemented by a sequence of edge reversals. Let us indeed first suppose that there is a vertex i in the cycle that has at least one degree of freedom. In that case, the cycle reversal is just a particular case of the path reversal, with $i = j$. We now assume that no vertex in the cycle has a degree of freedom. Let l be a vertex in the cycle, and m a vertex that does not belong to the cycle but has a degree of freedom. By Lemma 3, it follows that there exists a directed path from l to m. Let i be the last vertex in this path belonging to the cycle. There is trivially a path P from i to m such that every other vertex of this path does not belong to the cycle. The implementation of a cycle reversal by three path reversals is then represented in Figure 13. One begins by reversing the path P into P' such that i acquires a degree of freedom. As explained above, the cycle can then be reversed since it is a particular case of path reversal, and finally, one reverses the path P' back to P such that the degree of freedom acquired by i is re-transmitted to m.

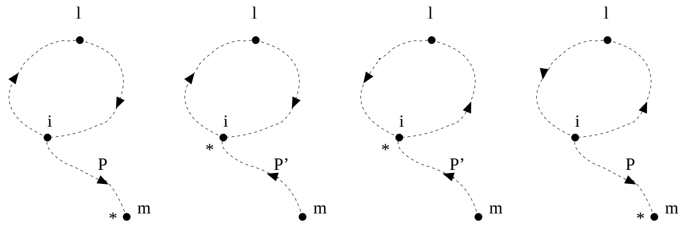

Fig. 13. Implementation of cycle reversal. The "*" represents one degree of freedom.

Remark 1: Both cycle reversal and path reversal are their auto-inverse, as is the case for edge reversal. Moreover, the fact that they can be implemented using only edge reversals is another way to show that they preserve minimal persistence.

It follows from Lemma 3 that one can arbitrarily reposition degrees of freedom using path reversals. The following result implies that two minimally persistent graphs having the same underlying undirected graph and the same positions for their degrees of freedom (all vertices having therefore the same out-degree in the two graphs) can differ only by cycles of opposite edges, and its proof provides a greedy algorithm to find such a cycle.

Lemma 4: Let $G_A = (V, E_A)$ and $G_B = (V, E_B)$ be two graphs having the same underlying undirected graph and such that every vertex has the same out-degree in both graphs. If an edge of G_A has the opposite direction to that in G_B, then it belongs to a cycle of such edges in G_A.

Proof: Suppose that $(i_0, i_1) \in E_A$ and $(i_1, i_0) \in E_B$ (i.e., this edge has opposite directions in G_A and G_B); then there exists at least one vertex $i_2 \neq i_0$ such that $(i_1, i_2) \in E_A$ and $(i_2, i_1) \in E_B$. For if the contrary holds, we would have $d^+(i_1, G_A) = d^+(i_1, G_B) - 1$, which contradicts our hypothesis. Repeating this argument recursively, we obtain an (infinite) sequence of vertices $i_0, i_1, i_2, \ldots$ such that for each $j \geq 0$, $(i_j, i_{j+1}) \in E_A$ and $(i_{j+1}, i_j) \in E_B$. Since there are only a finite number of vertices in V, at least one of them will appear twice in this sequence. By taking the subsequence of vertices (and induced edges) appearing in the infinite sequence between any two of its occurrences we obtain then a cycle of edges of G_A having opposite directions to those in G_B. This cycle does not necessarily contain (i_0, i_1). But if it does not, we can re-apply the same argument to G'_A, G'_B obtained from G_A and G_B by removing the edges of the cycle found. (i_0, i_1) has indeed an opposite direction in G'_A to that in G'_B, and these graphs satisfy the other hypotheses of the Lemma. Moreover, they contain less edges than G_A, G_B. Therefore by doing this recursively, we eventually obtain a cycle containing (i_0, i_1) since the number of edges in the graphs is finite.

■

D. Three primitive operations

Using the results of the two previous subsections, we can now show the following proposition.

Proposition 5: By applying a sequence of edge reversals to a given minimally persistent graph, it is possible to obtain any other minimally persistent graph having the same underlying undirected graph. Moreover, all the intermediate graphs are then minimally persistent.

Proof: Let G_A and G_B be two minimally persistent graphs having the same underlying undirected graph. Suppose that there is a vertex i which has less degrees of freedom in G_A than in G_B. Since at most three vertices have positive degree of freedom, there are at most three such vertices i. And since the total number of degrees of freedom is 3 in all minimally persistent graphs, there exists a vertex j which has more degree(s) of freedom in G_A than in G_B. In G_A i has thus necessarily less than two degrees of freedom and j as at least one degree of freedom. It follows then from Lemma 3 that there exists a directed path from i to j in G_A. The reversal of this path transfers a degree of freedom from j to i without affecting the number of degrees of freedom of the other vertices. Doing this at most two more times, the two graphs will have the same positions for their degrees of freedom.

We now show that that the following algorithm, which uses only cycle reversals, transforms then G_A into G_B:

```
**while** $\exists\ e$ having opposite direction in $G_A$ to that in $G_B$ **do**
    Select a cycle $C$ of such edges
    Reverse $C$ in $G_A$
**end do**
```

existence of C when $G_A \neq G_B$: This is a direct consequence of Lemma 4 since both graphs have the same underlying undirected graph and since all the vertices have the same out-degrees in both of them.

end of the algorithm: At each step of the loop, the number of edges having opposite directions in G_A and G_B is strictly reduced because all the edges for which directions are changed in G_A initially had an opposite direction in G_B (and because Proposition 1 forbids the presence of cycles of length 2 in a minimally persistent graph). Since there are only a finite number of edges, the algorithm finishes, and all the edges have then the same directions in both graphs.

The result of this proposition follows then from the fact that both path reversal and cycle reversal can be implemented by a sequence of edge reversals, which preserves minimal persistence.

■

From an autonomous agent formation perspective, suppose that a reorganization of the distance constraints distribution needs to be performed, and that this reorganization preserves the structure of constraints from an undirected point of view, i.e., the reorganization only involves changes of some directions. Proposition 5 implies that this can be done by a sequence of local degree of freedom transfers, in such a way that during all the intermediate stages, the formation shape is guaranteed to be maintained.

Let $\mathcal{T}$ be the set of operations containing vertex addition, edge splitting, and edge reversal. We can now state our main result, that every minimally persistent graph can be obtained by performing operations in $\mathcal{T}$ on an initial leader-follower seed.

Theorem 2: Every minimally persistent graph can be obtained by applying a sequence of operations in $\mathcal{T}$ to an initial leader-follower seed. Moreover, all the intermediate graphs are minimally persistent.

Proof: Consider a minimally persistent graph G. This graph is also minimally rigid. By Proposition 3, there exists thus a (possibly different) minimally persistent graph having the same underlying undirected graph that can be obtained by performing a sequence of operations in $\mathcal{S} \subset \mathcal{T}$ on an initial leader follower seed. By Proposition 5, G can then be obtained by applying a sequence of edge reversals on this last graph. Moreover, since all the operations in $\mathcal{T}$ preserve minimal persistence, all the intermediate graphs are minimally persistent.

■

To illustrate Theorem 2, consider the graph G represented in the right hand side of Figure 14(c), which is the graph of Figure 10 with $n = 2$. As explained in Section IV-B, it cannot be obtained by applying a vertex addition or an edge splitting on a smaller minimally persistent graph. However, by Theorem 2, it can be obtained by applying a sequence of operations in $\mathcal{T}$ on an initial leader-follower seed. Let us take 1 and 2 as respectively leader and follower of this initial seed. One can begin by adding 3, 4 and 5 using three vertex additions as shown in Figure 14(a). The graph obtained has the same underlying undirected graph as G, but the degrees of freedom are not allocated to the same vertices. By reversing the path $(5, 4, 2, 1)$ using a sequence of edge reversals, one can then transfer one degree of freedom from 1 to 5 as shown in Figure 14(b) such that in the obtained graph, all vertices have the same number of degrees of freedom (and therefore same out-degree) as in G. As stated in Lemma 4 , any edge of this graph that does not have the same direction as in G belongs to a cycle of such edges. The only such cycle here is C. By reversing it using a sequence of edge reversals, one

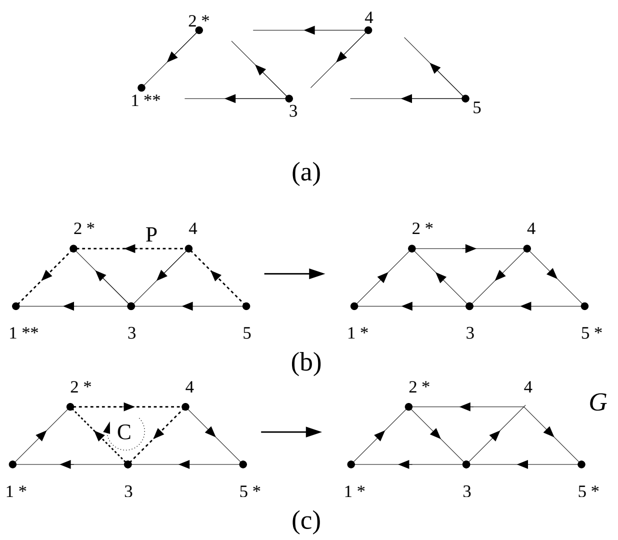

Fig. 14. Example of obtaining of a minimally persistent graph by applying a sequence of operations in $\mathcal{T}$ on a leader-follower seed. The graph G is obtained from the leader-follower seed by (a) three vertex additions, (b) the reversal of the path P and (c) of the cycle C.

finally obtains the graph G, as shown in Figure 14(c). Note that consistently with Theorem 2, all the intermediate graphs are minimally persistent.

Corollary 1: Every minimally persistent graph can be transformed into any other minimally persistent graph using only operations in $\mathcal{T} \cup \mathcal{T}^{-1}$.

Proof: Let G_A and G_B be two minimally persistent graphs. Since G_A can be obtained by applying a sequence of operations in $\mathcal{T}$ on a leader-follower pair, the leader-follower pair can be re-obtained from G_A by applying the reverse versions of these operations (which are all in $\mathcal{T}^{-1}$) in the reverse order. By Theorem 2 one can then obtain G_B from this leader-follower pair by a sequence of operations in $\mathcal{T}$. ■

The method proposed in the proof of Corollary 1 is generally not optimal in terms of the number of operations. Note also that unlike in the case of undirected Henneberg sequences, the number of operations to build a minimally persistent graph is not uniquely fixed by its number of vertices, although it is bounded in $O(|V|^2)$ as explained in [12] and [13].

Remark 2: Observe that the three operations in $\mathcal{T}$ are basic operations that can be performed locally. They can thus easily be implemented in a local way on an autonomous agent formation. It might however be possible to improve this basic character using for example an operations such as an *edge reorientation*, i.e., an operation consisting in changing the arrival vertex of an edge. As shown in Figure 15, a vertex addition operation and an edge reorientation operation can indeed implement an edge splitting operation which could thus be discarded. However, this would require an efficient and simple criterion to determine when such an edge reorientation operation can be performed, and no such criterion is presently available.

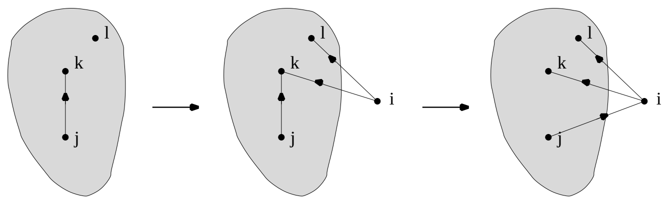

Fig. 15. Implementation of the edge splitting by a vertex addition and an edge reorientation. The vertex i is first added with two out-going edges by vertex addition, and the edge (j,k) is then reoriented and becomes (j,i).

E. Transforming a formation in a decentralized way

The graph depicting a formation's distance constraints is by essence a non-local concept. Therefore, reorganizing a formation into another formation requires a global view. However, several more local modifications can be done in a totally decentralized way.

Adding an agent by an operation corresponding to a vertex addition in the underlying graph is the simplest of those. It suffices for a new agent arriving in the neighborhood of a formation to sense two agents and to maintain its distances toward them constant. This agent can of course choose its relative position and its neighbors in such a way that its system of constraints is not ill-conditioned.

The edge splitting operation can also be performed in a decentralized way, as a combination of a vertex addition and an edge reorientation (see Figure 15). Suppose that an agent i has just joined a formation by means of a vertex addition and that another agent j has sensed this new presence and is willing to redirect one of its present edge (j,k) toward i, i.e. to maintain constant its distance toward i instead of its distance toward one of its present neighbors. j can ask i if k is one of its neighbors and redirect its constraints once it receives a positive answer. Note that this requires the ability for j and i to communicate and to identify (possibly temporarily) other agents in a unique and common way.

Edge reversal is an almost trivial operation if the agents have a 360° sensing range and if they are able to identify themselves in a unique way. It suffices for an agent i that is connected to an agent j having a degree of freedom to ask this agent j to actively maintain the distance between i and j. Suppose now that some agent i needs to increase its number of degrees of freedom. Lemma 3 guarantees the existence of a directed path from i to all vertices with positive number of degrees of freedom. Such a path can be found and can be reversed in a decentralized way using a depth-first research on the graph. One has however to be cautious to select only one degree of freedom to bring back to i and not all three of them. We propose the following simple decentralized algorithm to demonstrate how this problem can be solved. Real applications would require more advanced algorithms, in order for example to handle simultaneous concurrent demands or to choose "wisely" the degree of freedom to be transferred. This degree of freedom could for example be chosen according to a criterion such as its present localization, or numbers could be given to degrees of freedom to identify them in a unique way.

We suppose that the agent initially issuing a demand for a degree of freedom assigns a unique number to this demand $demand\_ID$, for example by juxtaposing its number $agent\_ID$ and its local present time. For all agents except the one issuing the demand, $received(demand\_ID)$ is initially set to $FALSE$. The agent issuing the demand just needs to apply the following procedure with its own identification number as second argument to one of its neighbors and then to the second one if the first demand is unsuccessful.

```
***agent.search_DOF***($demand\_ID$,$former\_ag\_ID$)

**if** $received(demand\_ID)$, **then** Return $FALSE$, STOP
**if** $\#neighbors < 2$, **then**
    Return $TRUE$,
    reverse $(former\_ag\_ID, agent\_ID)$,
    STOP
**end if**

**for** $i = 1 : 2$ **do**
    found = neighbor(i).search_DOF($demand\_ID$,$agent\_ID$)
    **if** found **then**
      Wait until $(agent\_ID, neighbor(i))$ reversed,
      Return $TRUE$,
      reverse edge $(former\_ag\_ID, agent\_ID)$,
      STOP
    **endif**
**end for**
**if** ∼found **then** Return FALSE
```

The algorithm should contain a procedure allowing an agent to cope with two different but simultaneous demands. Suppose that two agents ask i for a degree of freedom (the demands having different $demand\_ID$), i should have a way to decide for example to which one it provides the degree of freedom and to which agent it denies this service.

The problem of removing an agent is more intricate. An agent having out-degree 2 and in degree 0 can leave the formation without affecting its persistence, but this already requires each agent to know in which directed constraint it is involved. If the agent has an in-degree 1, it cannot leave the formation without warning the agent of which it is a neighbor and verifying if rigidity and persistence of the formation would be preserved after its departure. Since rigidity is a global notion, this cannot be easily done in a decentralized way.

VI. Conclusions and future work

In this paper, we have extended the Henneberg sequence concept to directed graphs using three operations allowing the building of all minimally persistent graphs. We also exposed some natural restrictions to these extensions, the main one being the impossibility of building all minimally persistent graphs using only confined generalized vertex additions or edge splittings. The existence of a set of confined operations not relying exclusively on Henneberg operations and allowing one to build all minimally persistent graphs remains however open. An improvement in the simplicity of the proposed operation set could come from the use of the edge reorientation, which would consist in changing the arrival point of an edge. However, the conditions under which minimal rigidity is preserved by this operation are not known yet.

From an autonomous agent point of view, our results provide a systematic approach to sequentially obtain or reorganize a minimally persistent agent formation. We explored briefly how such reorganization could be done in a decentralized way.

Finally, one of the main motivations for obtaining a directed version of Henneberg sequences was to develop tools for practical issues such as the merging of formations and the closing ranks problem. The undirected versions of these problems are indeed addressed using the undirected Henneberg theory. It remains now to see how our results can be efficiently used to address their directed version.

References

[1] J. Baillieul. *The Geometry of Sensor Information Utilization in Nonlinear Feedback Control of Vehicle Formations*, pages 1–24. Springer, 2005.

[2] J. Baillieul and A. Suri. Information patterns and hedging brockett's theorem in controlling vehicle formations. In *Proceedings of the 42nd IEEE Conf. on Decision and Control*, volume 1, pages 556–563, Hawaii, December 2003.

[3] J.M. Borky. Payload technologies and applications for uninhabited air vehicles (uavs). In *Proceedings of the IEEE Aerospace Conf.*, volume 3, pages 267–283, Aspen (CO), USA, February 1997.

[4] A. Das, J. Spletzer, V. Kumar, and C. Taylor. Ad hoc networks for localization and control. In *Proceedings of the 41st IEEE Conf. on Decision and Control*, volume 3, pages 2978–2983, Las Vegas, NV, 2002.

[5] K. Dogancay. Optimal receiver trajectories for scan-based radar localization. In *Proceedings Information, Decision and Control Conference 2007*, pages 88–93, Adelaide (SA), Australia, February 2007.

[6] T. Eren, B.D.O. Anderson, A.S. Morse, Whiteley W., and P.N. Belhumeur. Operations on rigid formations of autonomous agents. *Communication in Information and Systems*, pages 223–258, September 2004.

[7] T. Eren, B.D.O. Anderson, A.S. Morse, Whiteley W., and P.N. Belhumeur. Information structures to secure control of rigid formations with leader-follower structure. In *Proceedings of the American Control Conference*, pages 2966–2971, Portland, Oregon, June 2005.

[8] H.R Everett. *Sensors for Mobile Robots: Theory and Application*. A.K. Peters, 1995.

[9] B. Fidan and B.D.O. Anderson. Switching control for robust autonomous robot and vehicle platoon formation maintenance. In *Proceedings of the 15th Mediterranean Conference on Control and Automation*, Athens, Greece, June 2007.

[10] J.M. Hendrickx, B.D.O. Anderson, and V.D. Blondel. Rigidity and persistence of directed graphs. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2176–2181, Seville, Spain, December 2005.

[11] J.M. Hendrickx, B.D.O. Anderson, J.-C. Delvenne, and V.D. Blondel. Directed graphs for the analysis of rigidity and persistence in autonomous agents systems. *International Journal of Robust and Nonlinear Control*, 17:960–981.

[12] J.M. Hendrickx, B. Fidan, C. Yu, B.D.O. Anderson, and V.D. Blondel. Primitive operations for the construction and reorganization of minimally persistent formations. *Cesame research report 2006.62, arXiv:cs.MA/0609041 v1 8 Sep 2006.*

[13] J.M. Hendrickx, B. Fidan, C. Yu, B.D.O. Anderson, and V.D. Blondel. Elementary operations for the reorganization of minimally persistent formations. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS2006)*, pages 859–873, Kyoto, Japan, July 2006.

[14] L. Henneberg. Die graphische Statik der starren Systeme. Leipzig, 1911.

[15] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:331–340, 1970.

[16] S. Martinez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(3):661–668, 2006.

[17] R. Olfati-Saber and R.M Murray. Graph rigidity and distributed formation stabilization of multi-vehicle systems. In *Proceedings of the 41st Conference on Decision and Control*, volume 3, pages 2965–2971, Las Vegas, NV, December 2002.

[18] M. Pachter and J. Hebert. *Cooperative aircraft control for minimum radar exposure*, pages 199–211. Kluwer Academic, 2002.

[19] H.G. Tanner, G.J. Pappas, and V. Kumar. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 4(3):433–455, 2004.

[20] T. Tay and W. Whiteley. Generating isostatic frameworks. *Structural Topology*, (11):21–69, 1985.

[21] W. Whiteley. Some matroids from discrete applied geometry. In *Matroid theory (Seattle, WA, 1995)*, volume 197 of *Contemp. Math.*, pages 171–311. Amer. Math. Soc., Providence, RI, 1996.

[22] C. Yu, J.M. Hendrickx, B. Fidan, B.D.O. Anderson, and V.D. Blondel. Three and higher dimensional autonomous formations: Rigidity, persistence and structural persistence. *Automatica*, 43:387–402, March 2007.