

# Formation Reorganization by Primitive Operations on Directed Graphs

Julien M. Hendrickx, Barış Fidan, *Member, IEEE*, Changbin (Brad) Yu, *Member, IEEE*,  
Brian D. O. Anderson, *Life Fellow, IEEE*, and Vincent D. Blondel

**Abstract**—In this paper, we study the construction and transformation of 2-D 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 multiagent 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 online 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.

**Index Terms**—Graphs, multiagent formation, networks, persistence, rigidity, robots.

## 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.

Manuscript received March 15, 2007; revised July 17, 2007. Recommended by Associate Editor J. Lygeros. The works of J. M. Hendrickx and V. D. Blondel were 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. J. M. Hendrickx holds a F.R.S.-FNRS (Belgian Fund for Scientific Research) Fellowship. The work of B. Fidan, C. Yu, and B. D. O. Anderson is supported by the National Information and Communications Technology Australia (NICTA). NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. The work of C. Yu was supported by the Australian Government’s Department of Education, Science and Training.

J. M. Hendrickx and V. D. Blondel are with the Department of Mathematical Engineering, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium (e-mail: julien.hendrickx@uclouvain.be; vincent.blondel@uclouvain.be).

B. Fidan, C. Yu, and B. D. O. Anderson are with the Australian National University, Canberra, ACT 2601, Australia, and also with the National Information and Communications Technology Australia, Canberra, ACT 2601, Australia (e-mail: baris.fidan@nicta.com.au; brad.yu@nicta.com.au; brian.anderson@nicta.com.au).

Digital Object Identifier 10.1109/TAC.2008.920239

The results derived in this paper concern autonomous agents evolving in a 2-D space.

### A. Formations and Rigid Graphs

Many applications require the shape of a multiagent 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* interagent distances constant. In other words, some interagent distances are explicitly maintained constant so that all the interagent distances remain constant. The information structure arising from such a system can be efficiently modeled by a graph, where agents are abstracted by vertices and actively constrained interagent 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 interagent distances remain constant, as shown in Fig. 1. This property depends, indeed, almost only on the graph of distance constraints, and not on the particular agents positions and interagents 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 interagent 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 UAVs 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

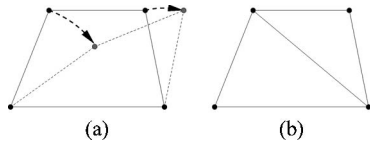


Fig. 1. Representation of nonrigid graph/formation in (a) and of a rigid one in (b). The solid structure in (a) can, indeed, be deformed to the dotted structure without breaking any distance constraint.

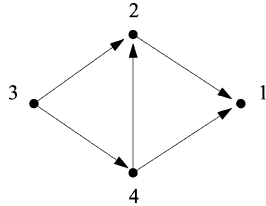


Fig. 2. Representation of a persistent graph, i.e., a rigid constraint consistent graph. This graph corresponds also to a leader–follower formation, where 1 is the leader and 2 the first follower.

this paper are working with agents in which optical sensors have blind 3-D cones. It can also be desired to ease the trajectory control of the formation, as it allows the so-called leader-follower formations [2], [7], [19]. In such a formation, one agent (leader) is free of interagent 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 Fig. 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 that 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 interagent distances remain constant, and, as a result, the formation shape is preserved. It is shown in [11] that persistence is actually the conjunction of

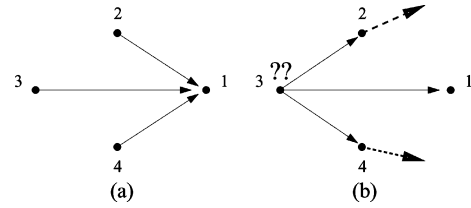


Fig. 3. Representation of a constraint consistent graph/formation in (a) and of a nonconstraint consistent one in (b), in 2-D. One can, indeed, see in (b) that for almost any uncoordinated continuous displacement of agents 2 and 4 (which are unconstrained), agent 3 is unable to move in such a way that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

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 Fig. 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 Fig. 2. For agents evolving in a 2-D 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.

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] and [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 2-D persistence, i.e., the persistence of graphs representing the information structure of a formation evolving in a 2-D space. Extension to the 3-D case may be difficult; even for undirected graphs, 3-D 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 nonconfined operations. Nonconfined 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 nonconfined 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] and [13], together with an alternative set of four operations allowing one to obtain all minimally persistent graphs.

## 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  $G$ .

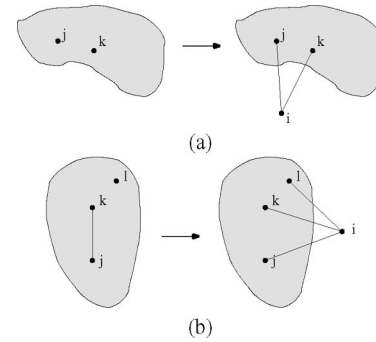


Fig. 4. Representation of the undirected vertex addition operation in (a) and of the undirected edge splitting operation in (b).

The intuitive meaning of the undirected notion of rigidity is explained in the Introduction. For a more formal definition, the reader is referred to [20] and [11]. In  $\mathbb{R}^2$ , 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  $E'' \subseteq E$ ,  $E'' \neq \emptyset$ , 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 of adding a vertex  $i$ , and connecting it to  $j$  and  $k$ , as shown in Fig. 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 of removing this edge, adding a vertex  $i$  and connecting it to  $j, k$ , and  $l$ , as shown in Fig. 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 of removing  $i$  and adding one edge among  $(j, k)$ ,  $(k, l)$ , and  $(l, j)$  in such a way that the 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 Fig. 5.

A *Henneberg sequence* is a sequence of graphs  $G_2, G_3, \dots, G_{|V|}$  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:* Every minimally rigid graph on more than one vertex can be obtained as the result of a Henneberg sequence [20]. Moreover, all intermediate graphs of such a sequence are minimally rigid.

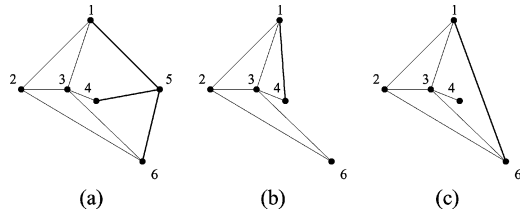


Fig. 5. Example of unfortunate added edge selection in reverse edge splitting. After the removal of vertex 5 from the minimally rigid graph (a), minimal rigidity can be preserved by the addition of the edge (1, 4), but not of (1, 6), as shown, respectively, on (b) and (c). In the latter case, the subgraph 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.

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

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 other positions are fixed. It, thus, represents in some sense the decision power of this agent. In a 2-D 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 that there are indeed three degrees of freedom to choose the position and orientation of a rigid body in a 2-D 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 an 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 sequence would be helpful 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 Fig. 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  $\hat{j}ik$  becomes too small, as represented in Fig. 7. Due to a shape modification, an initially sound formation can become ill-conditioned, as in Fig. 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 reimprove the conditioning of the formation systems without modifying the relative positions of the agents, as presented in Fig. 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

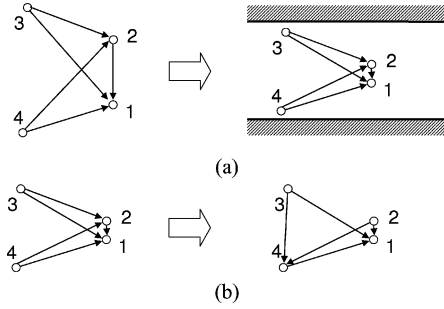


Fig. 6. (a) Formation needing to reduce the width of its shape in order to go through a narrow passage. As a result, the distance constraints sets of agents 3 and 4 become ill-conditioned. This can be solved by reorganizing the structure of constraints without modifying formation shape (b).

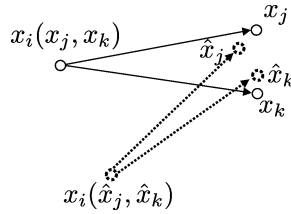


Fig. 7. Representation of an agent with an ill-conditioned set of constraints. Small positions variations or measurement errors  $\hat{x}_j - x_j$ ,  $\hat{x}_k - x_k$  can cause large modifications of the desired position of  $i$ .

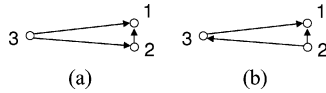


Fig. 8. Representation of two minimally persistent formations having the same undirected constraints. However, (b) is much better conditioned than (a) as the smallest angle between two constraints for which the same agent is responsible is much larger than in (a).

structure of the distance constraints. This could be the case for simple control reasons, as in the example represented in Fig. 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 of adding a vertex  $i$  and two directed edges  $(i, j)$  and  $(i, k)$ , as shown in Fig. 9(a). A reverse (directed) vertex addition consists of 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

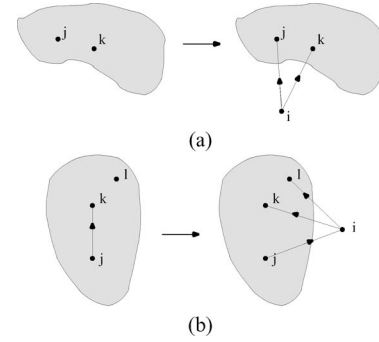


Fig. 9. Representation of (a) directed vertex addition and (b) edge splitting.

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 of adding a vertex  $i$ , an edge  $(i, l)$ , and replacing the edge  $(j, k)$  by  $(j, i)$  and  $(i, k)$ , as shown in Fig. 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 of 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 of 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

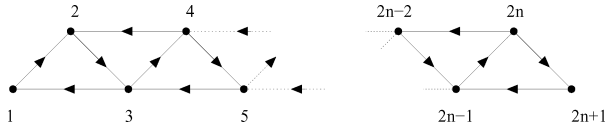


Fig. 10. Class of graphs on which no reverse vertex addition or edge splitting can be performed.

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 an 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. Consider the graph in Fig. 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 reobtain 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 of 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 earlier, 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 the 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 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 that 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 of 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 graphs by applying a sequence of confined generalized vertex additions 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 nonconfined edge splitting.

*Proof:* Suppose that one wants to remove a vertex without losing persistence from the provably minimally persistent graph represented in Fig. 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

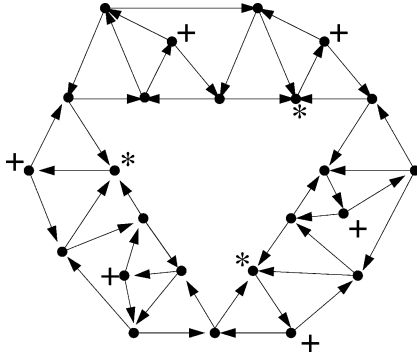


Fig. 11. Minimally persistent graph, no vertex of which can be removed without losing persistence by a reverse (generalized) vertex addition or a confined (generalized) reverse edge splitting. The symbol “\*” represents one degree of freedom. Vertices that are candidates to be removed by a reverse generalized edge splitting are labelled “+”.

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 *nonconfined* reverse generalized edge splitting.

Such a set of operations containing nonconfined 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. PURELY DIRECTED OPERATION

We have shown that unless we use operations that are not equivalent to Henneberg sequence operations from an undirected point of view, the use of nonconfined operations is required to be able to build all minimally persistent graphs. We, therefore, now introduce the edge reversal operation, the simplest possible nonconfined operation, which is neutral from an undirected point of view, as it only reverses the direction of one edge. We, then, define two macrooperations that 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.

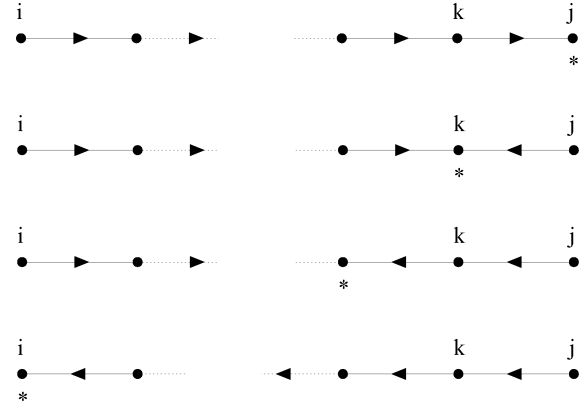


Fig. 12. Implementation of the path reversal by a sequence of edge reversals. The symbol “\*” represents one degree of freedom.

### 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 of 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 autoinverse 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, a *path reversal* consists of 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 Fig. 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.

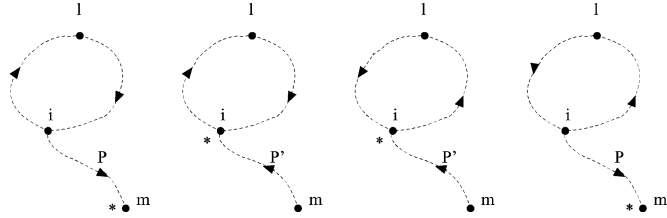


Fig. 13. Implementation of cycle reversal. The “\*” represents one degree of freedom.

*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 of 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 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 Fig. 13. One begins by reversing the path  $P$  into  $P'$  such that  $i$  acquires a degree of freedom. As explained earlier, 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 retransmitted to  $m$ .

*Remark 1:* Both cycle reversal and path reversal are their autoinverse, 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 a (infinite) sequence of vertices  $i_0, i_1, i_2, \dots$  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 reapply 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  $i$  that 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  $j$  that 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  $j$  has 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 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



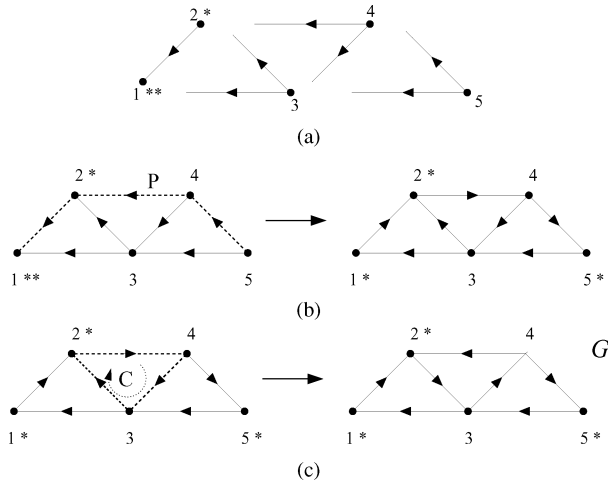


Fig. 14. Example of obtaining a minimally persistent graph by applying a sequence of operations in  $\mathcal{T}$  on a leader-follower seed. The graph  $G$  is obtained from the leader-follower seed by successively performing three vertex additions (a), reversing the path  $P$  (b), and reversing the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

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 Fig. 14(c), which is the graph of Fig. 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 Fig. 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 Fig. 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 finally obtains the graph  $G$ , as shown in Fig. 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 reobtained 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 operation such as an *edge reorientation*, i.e., an operation consisting of changing the arrival vertex of an edge. As shown in Fig. 15, a vertex addition operation and an edge reorientation operation can indeed implement an edge splitting operation that 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.

### E. Transforming a Formation in a Decentralized Way

The graph depicting a formation's distance constraints is, by essence, a nonlocal concept. Therefore, reorganizing a formation into another formation requires a global view. However,

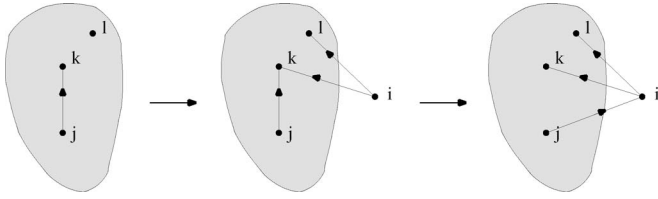


Fig. 15. Implementation of the edge splitting by a vertex addition and an edge reorientation. The vertex  $i$  is first added with two out-going edges by vertex addition, and the edge  $(j, k)$  is then reoriented and becomes  $(j, i)$ .

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 Fig. 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 edges  $(j, k)$  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  $360^\circ$  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**  $\sim$ 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. CONCLUSION 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 of changing the arrival point of an edge. However, the conditions under which minimal rigidity is preserved by this operation are not yet known.

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*. New York: Springer-Verlag, 2005, pp. 1–24.
- [2] J. Baillieul and A. Suri, "Information patterns and hedging brockett's theorem in controlling vehicle formations," in *Proc. 42nd IEEE Conf. Decision Control*, Hawaii, Dec. 2003, vol. 1, pp. 556–563.
- [3] J. M. Borky, "Payload technologies and applications for uninhabited air vehicles (UAVs)," in *Proc. IEEE Aerosp. Conf.*, Aspen, CO, Feb. 1997, vol. 3, pp. 267–283.
- [4] A. Das, J. Spletzer, V. Kumar, and C. Taylor, "Ad hoc networks for localization and control," in *Proc. 41st IEEE Conf. Decision and Control*, Las Vegas, NV, 2002, vol. 3, pp. 2978–2983.
- [5] K. Dogancay, "Optimal receiver trajectories for scan-based radar localization," in *Proc. Inf., Decision Control Conf. 2007*, Adelaide, Australia, pp. 88–93.
- [6] T. Eren, B. D. O. Anderson, A. S. Morse, W. Whiteley, and P. N. Belhumeur, "Operations on rigid formations of autonomous agents," *Commun. Inf. Syst.*, vol. 3, no. 4, pp. 223–258, Sep. 2004.
- [7] T. Eren, B. D. O. Anderson, A. S. Morse, W. Whiteley, and P. N. Belhumeur, "Information structures to secure control of rigid formations with leader-follower structure," in *Proc. Amer. Control Conf.*, Portland, OR, Jun. 2005, pp. 2966–2971.
- [8] H. R. Everett, *Sensors for Mobile Robots: Theory and Application*. Wellesley, MA: A.K. Peters, 1995.
- [9] B. Fidan and B. D. O. Anderson, "Switching control for robust autonomous robot and vehicle platoon formation maintenance," in *Proc. 15th Mediterranean Conf. Control Automat.*, Athens, Greece, Jun. 2007, pp. 1–6.
- [10] J. M. Hendrickx, B. D. O. Anderson, and V. D. Blondel, "Rigidity and persistence of directed graphs," in *Proc. 44th IEEE Conf. Decision Control*, Seville, Spain, Dec. 2005, pp. 2176–2181.
- [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," *Int. J. Robust Nonlinear Control*, vol. 17, pp. 960–981, 2007.
- [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 Res. Rep. 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 *Proc. 17th Int. Symp. Math. Theory Netw. Syst. (MTNS2006)*, Kyoto, Japan, 2006, pp. 859–873.
- [14] L. Henneberg, "Die graphische Statik der starren Systeme, Leipzig," 1911.
- [15] G. Laman, "On graphs and rigidity of plane skeletal structures," *J. Eng. Math.*, vol. 4, pp. 331–340, 1970.
- [16] S. Martinez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, no. 3, pp. 661–668, 2006.
- [17] R. Olfati-Saber and R. M. Murray, "Graph rigidity and distributed formation stabilization of multi-vehicle systems," in *Proc. 41st Conf. Decision Control*, Las Vegas, NV, Dec. 2002, vol. 3, pp. 2965–2971.
- [18] M. Pachter and J. Hebert, *Cooperative Aircraft Control for Minimum Radar Exposure*. Norwell, MA: Kluwer, 2002, pp. 199–211.
- [19] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," *IEEE Trans. Robot. Automat.*, vol. 4, no. 3, pp. 433–455, Jun. 2004.
- [20] T. Tay and W. Whiteley, "Generating isostatic frameworks," *Struct. Topology*, vol. 11, pp. 21–69, 1985.
- [21] W. Whiteley, "Some matroids from discrete applied geometry," in *Matroid Theory (Seattle, WA, 1995) of Contemp. Math.*, Amer. Math. Soc., Providence RI, 1996, vol. 197, pp. 171–311.
- [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*, vol. 43, pp. 387–402, Mar. 2007.



**Julien M. Hendrickx** received the Engineering degree in applied mathematics (with highest honors) and the Ph.D. degree in mathematical engineering from the Université catholique de Louvain, Louvain-la-Neuve, Belgium, in 2004 and 2008, respectively, under the supervision of Prof. Blondel.

He was a Visiting Researcher at the University of Illinois, Urbana Champaign, in 2003 and 2004, at National Information and Communications Technology (ICT) Australia, in 2005 and 2006, and at the Massachusetts Institute of Technology in 2006 and 2008. He is currently with the Department of Mathematical Engineering, Université catholique de Louvain. His main current research interests include multi-agent systems, swarming processes, decentralized control, graph and random graphs theory.



**Barış Fidan** received B.S. degrees in electrical engineering and mathematics from the Middle East Technical University, Ankara, Turkey, in 1996, the M.S. degree in electrical engineering from Bilkent University, Ankara, Turkey, in 1998, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 2003.

Since 2005, he has been with the National Information and Communications Technology Australia, Canberra, Australia, and the Research School of Information Sciences and Engineering, Australian National University, Canberra, where he is currently a Senior Researcher. His current research interests include autonomous formations, sensor networks, adaptive and nonlinear control, switching and hybrid systems, mechatronics, and various control applications.



**Changbin (Brad) Yu** (S'05–M'08) was born in Shandong, China. He received the B. Eng. degree in computer engineering (with first class honors) from Nanyang Technological University (NTU), Singapore, in 2004, and the Ph.D. degree in information engineering from the Australian National University (ANU), Canberra, Australia, in 2008.

He is currently a Researcher at the National Information and Communications Technology Australia, Canberra, Australia. He was a Visitor at the Université catholique de Louvain during 2005, at Yale University during 2006 and 2007, and at the National Institute of Informatics, Japan, in 2008. His current research interests include autonomous formations, multi-agents systems, sensor networks, and graph theory.

Dr. Yu is the recipient of the 2008 ARC Australian Postdoctoral Fellowship, the 2006 Chinese Government Award for Outstanding Students Abroad, the 2005 Australian Government's Endeavour Asia Award, and the Undergraduate Scholarship from the Republic of Singapore.



**Brian D. O. Anderson** (M'66–SM'74–F'75–LF'07) was born in Sydney, Australia. He received degrees in mathematics and electrical engineering from Sydney University, Sydney, N.S.W., Australia, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1966.

He is currently a Distinguished Professor at the Australian National University, Canberra, Australia, and also a Distinguished Researcher at National Information and Communications Technology (ICT) Australia. His current research interests include de-

centralized control and adaptive control.

Dr. Anderson was the recipient of the 1992 Bode Prize of the IEEE Control System Society, the 1997 IEEE Control Systems Award, and the 2001 IEEE James H. Mulligan, Jr., Education Medal. He was the President of the International Federation of Automatic Control (IFAC) from 1990 to 1993 and the Australian Academy of Sciences from 1998 to 2002. He is a Fellow of IFAC and of the Royal Society, and a Foreign Associate of the United States National Academy of Engineering.



**Vincent D. Blondel** received the M.Sc. degree in mathematics from Imperial College, London, U.K., in 1990 and the Ph.D. degree in applied mathematics from the Université catholique de Louvain, Louvain-la-Neuve, Belgium, in 1992.

He was a Visiting Researcher at the Royal Institute of Technology, Stockholm, Sweden, and at the Institut National de Recherche en Informatique et en Automatique (INRIA), Rocquencourt, France. During 2005–2006, he was an Invited Professor and a Fulbright Scholar at Massachusetts Institute of Tech-

nology, Cambridge. He is currently a Professor and Department Head at the Université catholique de Louvain, Louvain-la-Neuve, Belgium.

Dr. Blondel was the recipient of the Prize Wetrems of the Belgian Royal Academy of Science, the Society for Industrial and Applied Mathematics (SIAM) Prize on Control and Systems Theory, and the Ruberti Prize in Systems and Control of the IEEE in 2006.