Biconnecting a Network of Mobile Robots using Virtual Angular Forces

Arnaud Casteigts[1], Jérémie Albert[2], Serge Chaumette[2], Amiya Nayak[1], and Ivan Stojmenović[1]

[1]SITE, University of Ottawa, Canada
{casteig,anayak,ivan}@site.uottawa.ca

[2]LaBRI, University of Bordeaux, France
{albert,chaumett}@labri.fr

Abstract—This paper proposes a new solution to the problem of self-deploying a network of wireless mobile robots with simultaneous consideration to several criteria, that are, the fault-tolerance (*biconnectivity*) of the resulting network, its *coverage*, its *diameter*, and the *quantity of movement* required to complete the deployment. These criteria have already been addressed individually in previous works, but we propose here an elegant solution to address all of them at once. Our approach is based on combining two complementary sets of virtual forces: *spring* forces, whose properties are well known to provide optimal coverage at reasonable movement cost, and *angular* forces, a new type of force proposed here whose effect is to rotate two *angularly consecutive* neighbors toward one another when the corresponding angle is larger than $60°$ (even if these nodes are not direct neighbors). Angular forces have the global effect of biconnecting the network and reducing its diameter, while not affecting the benefits obtained by spring forces on coverage. In this paper we give a detailed description of the combination of both types of forces. We also provide an implementation relying only on position exchanges within two hops. Simulations results are finally presented to evaluate our solution with respect to the four considered criteria (coverage, biconnectivity, quantity of movements, and diameter), and compare it with prior approaches.

I. INTRODUCTION AND RELATED WORK

This paper addresses the problem of self-deploying a swarm of wireless mobile robots in a biconnected fashion. The motivations for deploying robotic sensor networks in general include accessing places where human cannot go (*e.g.* remote planets, underwater area, dangerous spots with chemical or radioactive leaks), or automating fastidious large scale deployment (*e.g.* spreading intrusion or fire detectors over a large area). Once deployed, the robots are intended to perform collective tasks related to the monitoring, collection, or processing of information sensed in the surrounding environment.

The efficiency and reliability of such networks depend on several criteria. To be fault-tolerant (tolerate the failure of any single robot), the resulting network must be *biconnected*. A network is biconnected if it remains connected after excluding any of its nodes from the network. Besides fault-tolerance, it is important to *maximize the collective coverage* of the network (that is, the overall area the robots can sense from or serve), while minimizing the *network diameter* (longest shortest path between any pair of nodes) to ensure efficient communications. Finally, these criteria should not be satisfied at the expense of a too high energy consumption, and must thus be achieved with *moderate movements* by the robots.

The problem of maximizing the coverage of a network of robots has already been addressed in numerous works, using either virtual repulsion forces (or a combination of repulsion and attraction forces, called *spring forces*) [6–8, 12] or geometrical approaches [1, 8, 9] that equivalently regulate the inter-nodal distance, and as a side effect organize the robots as equilateral triangles (whose good properties with regards to coverage are well known). Interestingly, these approaches can indirectly create, or maintain, biconnectivity at places where the density of robots is already high. Yet, they do not achieve it otherwise.

Biconnectivity has been explicitly addressed in other works. In the most general context (targetting arbitrary, sparse, and even possibly *dis*connected topologies), solutions based on a *common reference point* toward which all robots can converge and biconnect were proposed in [6, 10, 11]. Having a common reference point known by all the robots makes the biconnectivity problem trivial to solve (at least, without obstacles), but is a fairly strong assumption. Without this assumption, other solutions have been proposed to biconnect networks that are already *connected* [2, 5], based on the movements of some selected robots, but these solutions do not consider the coverage nor the diameter of the resulting network. In addition, the solutions in [2] are centralized, and the algorithm in [5] fails to biconnect in numerous situations (which is explicitly conceded by the authors).

The solution we propose in this paper does not use a common reference point, and therefore does not handle *arbitrarily disconnected* topologies. It addresses however any kind of initially *connected* topology (regardless of whether released as a high density conglomerate or as a randomly distributed set of robots), and also deals with initially *disconnected* topologies in closed areas when the number of robots is sufficient to allow repulsion forces to make them connect with each other (this feature can be found in any repulsion-based solution). Besides obtaining a much higher success rate than [5], the main novelty is to address all the criteria described above at once. Its principle is based on combining spring forces with a new kind of force called *angular* forces. Whereas spring forces determine the distances between nodes and naturally form equilateral triangles at dense places, angular forces strive to reduce the angles formed by pairs of *angularly consecutive* neighbors of a same node regardless of whether these neighbors are already in range of each other (so that equilateral triangles are also formed at sparse places). At a global scale, angular forces have the effect of biconnecting the network and reducing its diameter at the same time. The intuition of this behavior can be obtained by looking at Figure 1.
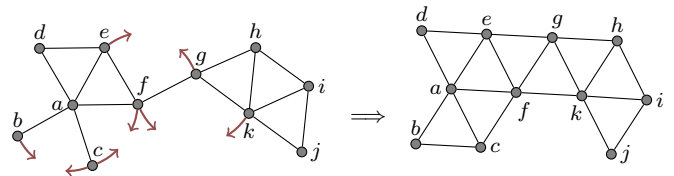

Fig. 1. Intuitive example of the effect of angular forces.

The paper is organized as follows. The assumptions, notations, and network model are given in Section II. Section III presents our solution and describe the technical challenges that had to be faced to successfully combine angular forces with spring forces (without negative interference between them). A possible implementation is then proposed in Section IV. Note that the principle of rotation can be implemented using at least two possible approaches, depending on what robot does what action (*e.g.* does a robot ask its neighbors to rotate, or do these neighbors take such decision alone). In the proposed

implementation all robots acquire (a subset of) the two-hop neighbors positions and then determine their movements alone. We then briefly discuss some of the simulation results we obtained in Section V, and eventually conclude with some remarks.

II. NETWORK MODEL, NOTATIONS, AND ASSUMPTIONS

We consider a network of autonomous mobile robots enabled with wireless communication and movement capabilities. We assume that each robot is able to acquire frequently its absolute position (the very principle of angular forces might however work with *relative* positions). Each robot has *communication* and *sensing* ranges that are distinct, but uniform among the swarm. The communication range, C_R, determines the distance up to which two robots can directly communicate. The sensing range, S_R, determines the circular area around a robot from which it can acquire information about the monitored environment. This is referred to as its *coverage*.

In the proposed implementation, robots are assumed to discover their neighborhood by means of periodic beacon exchanges that comprise position information. We assume that beacons support *piggybacking* (insertion of data in a dedicated field), up to 6 times the size of a position (in order to propagate two-hops information). No additional communication is used beyond the beacons. Finally, as the robots apply the algorithm, they regulate their inter-distance around a threshold d_{th}. Whereas no restrictions apply on C_R and S_R, we require $d_{th} \leq 0.851 \, C_R$ (for reasons developed in Section III-E).

III. THE PROPOSED VIRTUAL FORCE SCHEME

A. Representing individual forces

Individual forces are usually represented as two-dimensional polar coordinate vectors $\vec{F} = (direction, magnitude)$, with the strength of the force being directly encoded in the magnitude (*e.g.* as a linear, quadratic, or exponential expression of the quantity to correct). We represent forces differently, using a *three*-dimensional vector $\vec{F}$ containing a *direction*, a *magnitude*, and a separate *weight*, respectively noted $\theta_{\vec{F}}$, $|\vec{F}|$, and $\omega_{\vec{F}}$. This solution allows us to disassociate the magnitude of a force from its real strength (weight), and thus to use the magnitude to represent the exact distance to the force's equilibrium point.

B. Directions and Magnitudes

We describe now the computation of directions and magnitudes of individual force vectors, for both spring and angular forces. The way these forces are weighted and combined with each other is discussed later. Each spring force involves a pair of node (whose *inter-distance* is to be regulated), whereas an angular force involves three nodes: a center node and two of its angularly consecutive neighbors (whose *angle* is to be regulated).

1) Spring Forces: given two nodes i and j, and their inter-nodal distance d_{ij}, we call spring *correction* (and abbreviate *cor*) the difference between d_{ij} and the desired threshold d_{th}. So $cor = \frac{|d_{ij} - d_{th}|/2}{d_{th}}$. (This quantity is expressed as a ratio over d_{th} for independence to the unit.) The definition of spring forces is then

$$\vec{F}_{ji} = \begin{cases} (\theta_{\vec{ij}}, d_{th} \times cor, weight_S(cor)), & \text{if } d_{ij} > d_{th} \\ (\theta_{\vec{ji}}, d_{th} \times cor, weight_S(cor)), & \text{otherwise} \end{cases}$$

where $\theta_{\vec{ij}}$ is the orientation of the segment linking i to j, $weight_S()$ is a function defined hereafter, and $\vec{F}_{ji}$ can be read as *the spring force that j exerts on i*. Note that the same weight function is considered for both attraction and repulsion.

2) Angular Forces: for a given angle $\gamma = \widehat{abc}$, where a and c are two consecutive neighbors of b in the clockwise direction (see Figure 5(a) for an illustration), the magnitude of the force applying on a, noted $|\vec{F}_{bac}|$ ($\vec{F}_{bac}$ can be read as *the angular force that b exerts on a with respect to c.*), is 0 if $\gamma \leq 60°$, and $d_{th} \times tan(\beta)$ otherwise, where β, called the *correction angle*, is half the difference between the current angle and $60°$ ($\pi/3$ in radians). As for the direction, it simply corresponds to a physical *torque* applied to the two nodes, *i.e.,* $\pm 90°$ relative to the angle of the segment from the node to the center, depending on their relative position ($\theta_{ab} + 90°$ for a and $\theta_{cb} - 90°$ for c, in the example of Figure 5(a)).
*Remark:* we described here angular forces irrespective of their implementation. At least two approaches could be considered in practice: either node b computes the forces it exerts on a and c and sends them the corresponding vectors, or a and c compute these vectors themselves thanks to a 2-hop position knowledge they have on their neighborhood (this latter solution is the one we propose in Section IV).

C. Cases where forces apply

Both spring and angular forces do not systematically apply among all neighbors. There are two restrictions: one for both types of forces, and the second specific to angular forces.

1) Neighbors selection: in order for the robots to stabilize as equilateral triangles, the number of neighbors a robot can interact with should be limited to 6 (since $60° \times 6 = 360°$). The strategy proposed in [6] is to have some neighbors *shielding* the others (*i.e.,* making them not considered). More precisely, for a given node i, a neighbor k *shields* another neighbor k' if k is closer to i and the angle $\widehat{kik'}$ is smaller than $60°$. This scheme is illustrated on Figure 2(a) (where nodes s_1, s_2, s_3, and s_4 shield nodes e_1, e_2, e_3, e_4, and e_5).

We adopt the same principle but consider another value than $60°$. Indeed, this value does not allow to select more than 5 neighbors in practice (because selecting 6 neighbors would require that the angles between them are all *exactly* equal to $60°$). In order to maximize the possibility of selecting 6 neighbors while forbidding the selection of 7 neighbors, one might reduce the shielding angle down to $360/7°$ ($\simeq 51.43°$). In fact, we reduce it only to $54°$ because smaller values would allow pentagonal configurations to be stably maintained by spring forces (*e.g.* we wants a to dismiss c because of b, in Figure 2(b))
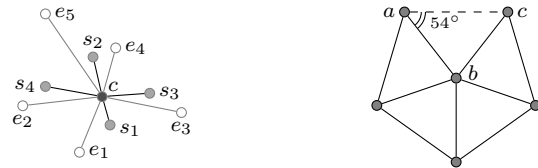
(a) Example of selection. (b) Minimal shielding angle w.r.t. pentagons.

Fig. 2. Neighbor selection.

2) Largest angle: given a node and n angles formed by its n selected neighbors, angular forces do not exert relatively to the largest angle if $n < 6$. (Looking again at the example of Figure 1, this causes e.g. the angle $\widehat{bad}$ to be ignored.)

D. Averaging the sums and moving

In every iteration, a given node can be subject to a number of spring forces and angular forces competing with each other. Determining how these vectors are combined to generate an effective movement is a crucial step in the solution.

Virtual force schemes usually combine several vectors by summing them. Keeping in mind that usual vectors are only two-dimensional (direction, magnitude), the problem with this approach is that if several vectors pull or push the node in a same direction, the magnitude of the resulting vector is accordingly increased, with the consequence that the node may be asked to move beyond the equilibrium position (then reverse its direction at the next iteration, and so forth), which generates unwanted oscillation movements. This motivates the choice of *averaging* the vectors instead of summing them.

Hence, given the set $\mathcal{F}$ of all forces exerting on a given node, each having the form (direction $\theta_{\vec{F}}$, magnitude $|\vec{F}|$, and weight $\omega_{\vec{F}}$)), we combine them by means of a *weighted average*. This weighted average reduces the set of all force vectors into a single two-dimensional *distance* vector (in Cartesian coordinates of unit d_{th}), hereafter called the *resulting vector*, as follows

$$\vec{V}_{res} = \left(\frac{\sum_{\vec{F} \in \mathcal{F}} (cos(\theta_{\vec{F}}) \times |\vec{F}| \times \omega_{\vec{F}})}{\sum_{\vec{F} \in \mathcal{F}} \omega_{\vec{F}}}, \frac{\sum_{\vec{F} \in \mathcal{F}} (sin(\theta_{\vec{F}}) \times |\vec{F}| \times \omega_{\vec{F}})}{\sum_{\vec{F} \in \mathcal{F}} \omega_{\vec{F}}} \right).$$

The contribution of a given force vector to the resulting vector is now directly proportional to the product of its *magnitude* by its *weight*. Therefore, in order to set the weights function properly, we have to consider their impact through this particular product, hereafter referred to as *strength*.

E. Weights

At places with sufficient densities, spring forces alone suffice to build equilateral triangles. Angular forces should preferably not interfere with this. In fact, it would be convenient to consider any equilateral triangle maintained by spring forces as an *unbreakable compound*, and limit the role of angular forces to form new such structures or to rotate existing structures toward one another in order to form larger compounds (as in the case of Figure 1).

This strategy dictates a set of constraints. First, spring forces must have priority over angular forces, that is, angular forces must be neglected as long as spring forces are not close to their equilibrium, and then take over progressively (in physical terms, we could call it an *asymptotic freedom* of angular forces with respect to spring forces). We achieve this by weighting spring forces exponentially ($weight_S(cor) = exp(cor)$), while maintaining angular forces below a given threshold (this priority relation is illustrated on Figure 3).
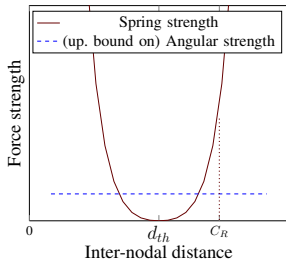

Angular forces must prevail only when spring forces are nearly satisfied. An additional constraint is to ensure that the angular strength is always significantly lower than the spring strength at $d_{th}/0.851$ (minimal C_R), so that angular forces cannot disconnect the network in case of conflicting configuration.

Fig. 3.   Priority between forces

Before discussing more precisely the upper limit of angular forces, let us first focus on their independent behavior with

respect to the correction angle. A specific requirement is that angular forces get stronger as the correction angle *decreases*. The reason for this counter-intuitive design choice is that local configurations with several competing outcomes must not stabilize in an even state. An example of such even configuration is given in Figure 4. Hence, the strength of the angular forces (that is, the product $magnitude \times weight$) must increase as the correction decreases, still being equal to 0 when the correction is 0 (angle of 60°), and finally be continuous over all its definition domain, from 0 to $\pi/3$ (the correction at 180° is $\pi/3$), which generates an apparent contradiction.
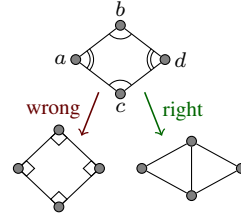
If the angular strength was designed to *increase* with the correction angle, this example configuration would become a square and stabilize as such, which is not a desired outcome. Now, if the strength *decreases* with the correction angle, the candidate outcome that has the biggest advantage will increase its advantage over time (here, b and c will join and form two new equilateral triangles).

Fig. 4.   Example of competition between two configurations.

In fact, the angular strength does not need to be increased until the correction angle is 0, since the two rotating nodes will select each other for spring attraction before this happens. Looking at Figure 5(b), this mutual selection will occur as soon as a stops shielding both nodes from one another, which corresponds to angles $\widehat{acb}$ and $\widehat{bac}$ being $< 54°$ (the shielding angle). The angular strength exerted by a on b and c (and more generally by the center node to its two considered neighbors) can thus start decrease once the local angle passes below 72° (see Figure 5(b)), which corresponds to a correction of 6° ($\pi/30$ radians). Note that guaranteeing that b and c will be within range of each other when this happens is the reason why we require $d_{th} < 0.851\,C_R$.

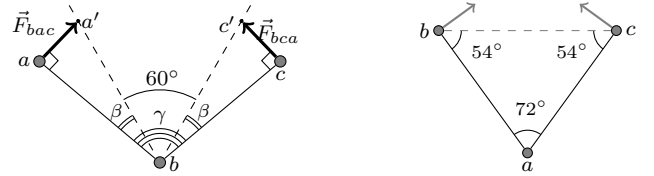(a) Rotation with respect to b (b) Configuration of maximal strength

Fig. 5.   Illustration pictures for angular forces.

As a conclusion, the *strength* of angular forces must increase as the correction decreases from $\pi/3$ to $\pi/30$, then decrease to 0 as the correction decreases from $\pi/30$ to 0, and still be continuous (these constraints can be visualized by glancing at the plot of the strength in Figure 6). One possible way to obtain this behavior is to define the angular weight as a negative exponential with the correction β as variable and specific slope parameters. We pose $weight(\beta) = e^{-a\beta^b}$. The strength ($magnitude \times weight$) is thus equal to the product

$$strength(\beta) = d_{th} \times tan(\beta) \times e^{-a\beta^b} \tag{1}$$

An infinity of pairs (a, b) satisfy the required constraints, each one leading to a different decrease rate between $\pi/30$ and $\pi/3$. For example using $b = 1$, the decreasing rate is such that the ratio between $strength(\pi/30)$ and $strength(\pi/3)$ is more than 8500. Knowing that angular forces are already bounded by design with respect to spring forces, such a slope will prevent the robots from rotating efficiently at large angles. We arbitrarily set b to 0.5 (that is, a *square root*), which offers a much slower

decrease rate, and then set a so as to shift the maximum of the strength at $\pi/30$. This computation (of a knowing b) can be done using the derivative of the strength by finding what value of a leads to a zero at $\pi/30$. This leads to

$$a = \frac{\sqrt{30\pi} \times (tan^2(\frac{\pi}{30}) + 1)}{15 \times tan(\frac{\pi}{30})} \simeq 6.222 \qquad (2)$$

which gives a (preliminary) weight of $e^{-6.222\sqrt{\beta}}$. The shape of this weight is illustrated on Figure 6. Note that the ratio between $strength(\pi/30)$ and $strength(\pi/3)$ is now ~ 4.73.
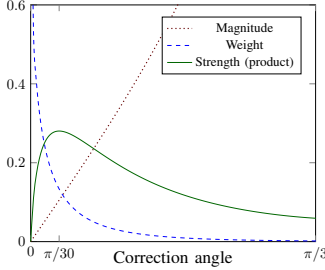

Fig. 6. Strength of angular forces (*the plot of the strength is increased 20 times for visibility purpose*).

After the shape of angular forces is determined, the last step consists in raising it by a multiplicative factor, up to the highest possible value at which they do not interfere negatively with spring forces. In the same way as the *shielding angle* of neighbors selection was limited in order to avoid stable pentagonal configurations (see Section III-C1), we limit here the multiplicative factor so as to prevent *heptagonal* (or higher order polygonal) configurations (as explained in Figure 7).
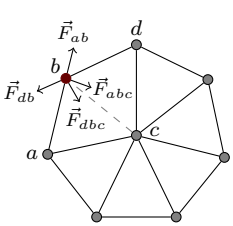
Due to the neighbors selection scheme, at least one node, b on the figure, will be discarded by c, and will therefore not interact with it. The desirable behavior here would be that the spring repulsion from a and d ($\vec{F}_{ab}$ and $\vec{F}_{db}$) push b away in the direction $\vec{\theta_{cb}}$. However, a and d also exert an angular force on b that tends to maintain it in place ($\vec{F}_{abc}$ and $\vec{F}_{dbc}$). Whether b will be successfully ejected thus depends on the relative strength between spring forces and angular forces.

Fig. 7. Breaking heptagonal configurations.

The analytical characterization of the multiplicative factor is left open for future work. We determined it experimentally by forming such an heptagon and multiplying angular forces by a high factor, then lowering the factor progressively until the heptagon breaks by itself, which occurred at a factor ~ 1.15. The final formula for the weight of angular forces is thus

$$weight_A(\beta) = 1.15 \times e^{-6.222\sqrt{\beta}} \qquad (3)$$

where β is the angular correction in radians. Finally, we can notice that the resulting strength satisfies the constraint mentioned in Figure 3 (angular forces cannot prevail over spring attraction to disconnect the network). In fact, we precisely have $strength_S(correction(d_{th}/0.851)) \simeq 5.92 \times strength_A(\pi/30)$.

F. Friction force

We consider a threshold on $|\vec{V}_{res}|$ below which the robots do not move. This threshold, noted *epsilon*, can be as small as desired and essentially allows to regulate the priority between the quantity of movements consummed and the other metrics (biconnectivity, diameter, coverage), as discussed in Section V.

IV. IMPLEMENTATION

We propose an implementation based on a periodical exchange of beacons conveying two-hops position information. More precisely, each beacon contains the (future) position of its emitter, along with the (expected) position of this emitter's selected neighbors. Upon reception, beacons are stored in a *mailbox*, which is read offline at regular intervals t_{rnd} (rounds).

Informally, the algorithm is as follows. In each round, robots starts by reading the messages received during the previous round, deduce their new list of neighbors and update local variables to store the corresponding information (positions of the neighbors and positions of the neighbors' selected neighbors). Based on this information, robots first determine their own selection, then compute their next position based on forces virtually exerted by the neighborhood (over 1-hop for spring forces, 2-hops for angular forces). They then send the next beacon including their next position and the (expected) positions of their selected neighbors, and finally start to move toward the next position. The detailed process is given in Algorithm 1.

Algorithm 1 Baseline algorithm (runs every t units of time)

1: $neighbors[] \leftarrow \emptyset$
2: $messages[] \leftarrow mailbox.popAllMessages()$ // *c*lears the mailbox.
3: **for all** $msg \in messages[]$ **do**
4:      $neighbors[] \leftarrow neighbors[] \cup msg.sender$
5:      $neighbors[msg.sender].position \leftarrow msg.pos$
6:      $neighbors[msg.sender].selection[] \leftarrow msg.selection[]$
7: **end for**
8: $me.selection[] \leftarrow select(me, neighbors[])$
9: $nextP \leftarrow computeNextPosition(me)$
10: $send(newMsg(sender=me, pos=nextP, selection=me.selection[]))$
11: **if** $nextP \neq currentPos$ **then**
12:      $moveTo(nextP)$
13: **end if**

The function $computeNextPosition$ is given by Algorithm 2, where $getSpringForce(ng, me)$ returns the force $\vec{F}_{ng\,me}$, and $getAngularForce(ng, me)$ returns an 2-elements array containing $\vec{F}_{ng\,me\,p}$ and $\vec{F}_{ng\,me\,s}$ (where p and s are the predecessor and successor of me in $ng.selection$, respectively).

Algorithm 2 $computeNextPosition(me)$

1: $List[] \leftarrow \emptyset$
2: **for all** $ng \in me.selection[]$ **do**
3:      **if** $me \in ng.selection[]$ **then**
4:          $List[] \leftarrow List[] \cup getSpringForce(ng, me)$
5:          $List[] \leftarrow List[] \cup getAngularForce(ng, me)$
6:      **end if**
7: **end for**
8: $\vec{V}_{res} \leftarrow getWeightedAverage(List)$
9: $d = min(|\vec{V}_{res}| \times d_{th}, d_{max})$
10: **if** $d \leq epsilon$ **then**
11:      $d \leftarrow 0$
12: **end if**
13: $\vec{V}_{res} \leftarrow (\theta_{\vec{V}_{res}}, d)$ // $\vec{V}_{res}$ *is truncated to a magn. of d (polar notation)*
14: **return** $current\_position + \vec{V}_{res}$

Our simulations assumed that robots can move at some speed v_{max} with instantaneous acceleration and direction change, that is, up to $d_{max} = v_{max} \times t_{rnd}$ per round. Note that the distance unit of $\vec{V}_{res}$ is still d_{th}. In each round, the robots will thus move $|\vec{V}_{res}| \times d_{th}$ or d_{max}, whichever is the shortest. The value chosen for d_{max} consequently have an impact on the number of rounds used to deploy. However, simulations showed that it has virtually no impact on the quantity of movements involved, nor on the outcome of the algorithm.
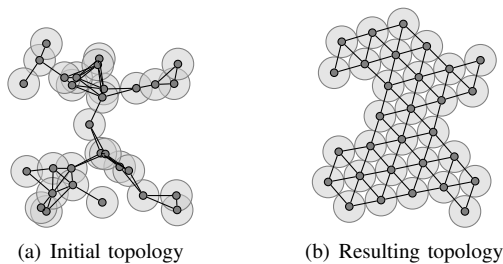
(a) Initial topology   (b) Resulting topology

Fig. 8. Example of arbitrarily connected topology of 30 nodes (and corresponding outcome with our algorithm).

V. EXPERIMENTATIONS

The algorithm proposed in this paper can possibly target three different contexts. These contexts are:

- All robots are released from a same place as a single high-density conglomerate (*Case A*);
- The robots are arbitrarily distributed in a closed area and do not necessarily form a connected topology. However, they are released in sufficient number to allow mutual repulsion to eventually make them connected (*Case B*);
- The robots are arbitrarily distributed in an open area, but their topology is already connected (*Case C*, see Figure 8).

At first sight, the benefits of using angular forces is not evident in the first two cases (*A* and *B*). We thus compared our solution to the use of spring forces *alone* in these contexts. As for case *C*, which is the one that mainly motivated this work (and for which spring forces alone are not adapted), we compared our solution to that of [5], the only known distributed solution addressing such scenarios.

Due to space limitation, only a brief summary of these simulations is now presented. In a nutshell, the combination of both kinds of force (our solution) prove relevant in Case A (using slightly less movements than spring forces alone, and maintaining the biconnectivity of the whole group whereas spring forces alone do not; however the combination induced a cost of ∼25% more time to stabilize). In Case B, our solution prove *not* relevant due to its requirement of having $d_{th} \leq 0.851\,C_R$. In Case *C*, where the topologies were generated by drawing the positions of nodes uniformly at random (with an appropriate density [3]), then selecting only the connected ones for simulations, our algorithm achieved biconnectivity in more than 90%, against 50% for the algorithm in [5]. It also led to 60% more coverage, and 8% less diameter (note that the same coverage can be observed in any algorithm having a *repulsion*-based mechanism). Finally, we observed that varying the threshold *epsilon* can be used to leverage the priority between *movements* and *biconnectivity*. All simulations were performed using the *JBotSim* platform [4].[1]

VI. CONCLUDING REMARKS

In this paper we investigated the joint use of virtual spring forces with a new kind of forces called *angular* forces, which have the effect of *contracting* large angles formed by nodes and pairs of neighbors. These forces have the global effect of biconnecting the network, at the same time as reducing its diameter. The paper presented a step by step thorough design of these forces, which was mostly led by physical

[1]A demo. is available at: http://jbotsim.sf.net/examples/bico.html

constraints that we identified and carefully discussed. Besides these contributions, we introduced more general design aspects, such as the clear separation between the weight of a force and its magnitude, as well as the technique of merging multiple forces by means of a *weighted average*, rather than summing. These approaches eliminates two systematic sources of oscillation.

The benefits of adding angular forces to spring forces were tested in three different contexts, namely i) robots released from the same place, ii) robots distributed at random within a close area (possibly disconnected), and iii) robots distributed at random, but connected, in an open area. Simulation results showed that the use of angular forces was not pertinent in case ii), whereas it was bringing substantial benefits in case i). Regarding case iii), which was the main motivation of this work, we compared the solution to the only known distributed (*i.e.,* non centralized) competitor. Where that algorithm was able to achieve biconnectivity in approximately 50% of the cases, ours did it in more than 90% of the cases up to 200 nodes (and 95% for less than 100 nodes).

Another aspect that simulations revealed about the algorithm is the fact that one specific parameter (*epsilon*, the threshold below which a robot do not move at all) could be highly instrumental in balancing the trade-off between the quantity of movements performed, and the ratio of biconnectivity achieved.

REFERENCES

[1] K. Akkaya and M. Younis. C2AP: Coverage-aware and connectivity-constrained actor positioning in wireless sensor and actor networks. In *IEEE International Performance, Computing, and Communications Conference (IPCCC'07)*, pages 281–288, New Orleans, Louisiana, 2007.
[2] P. Basu and J. Redi. Movement control algorithms for realization of fault-tolerant ad hoc robot networks. *IEEE Network*, 18(4):36–44, 2004.
[3] C. Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 80–91, New York, NY, USA, 2002. ACM.
[4] A. Casteigts. *The JBotSim Library*. e-Print (arXiv:1001.1435), Jan 2010. http://arxiv.org/abs/1001.1435.
[5] S. Das, H. Liu, A. Nayak, and I. Stojmenovic. A localized algorithm for bi-connectivity of connected mobile robots. *Telecommunication Systems*, 40:129–140, April 2009.
[6] M. Garetto, M. Gribaudo, C.-F. Chiasserini, and E. Leonardi. A distributed sensor relocation scheme for environmental control. *IEEE Intl. Conf. on Mobile Adhoc and Sensor Systems (MASS'07)*, pages 1–10, Oct. 2007.
[7] A. Howard, M. Mataric, and G. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *6th International Symposium on Distributed Autonomous Robotics Systems (DARS'02)*, Fukuoka, Japan, June 2002.
[8] G. Lee and N.Y. Chong. A geometric approach to deploying robot swarms. *Annals of Math. and Artif. Intelligence*, 52(2-4):257–280, 2008.
[9] M. Li, J. Harris, M. Chen, S. Mao, Y. Xiao, W. Read, and B. Prabhakaran. Architecture and protocol design for a pervasive robot swarm communication networks. *Wireless Comm. and Mobile Comp.*, 2009.
[10] H. Liu, X. Chu, Y.-W. Leung, and R. Du. An efficient physical model for movement control towards bi-connectivity in robotic sensor networks. *IEEE Journal on Selected Areas in Communications*, 2010. to appear.
[11] G. Tan, S.A. Jarvis, and A.-M. Kermarrec. Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks. In *28th International Conference on Distributed Computing Systems (ICDCS'08)*, pages 429–437, June 2008.
[12] Y. Zou and K. Chakrabarty. Sensor deployment and target localization based on virtual forces. In *IEEE 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*, volume 2, pages 1293–1303, March 2003.