Computer Communications xxx (2011) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Biconnecting a network of mobile robots using virtual angular forces ☆

Arnaud Casteigts [a,*], Jérémie Albert [b], Serge Chaumette [b], Amiya Nayak [a], Ivan Stojmenovic [a,c]

[a] SITE, University of Ottawa, Canada
[b] LaBRI, University of Bordeaux, France
[c] University of Novi Sad, Serbia

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Wireless mobile robots
Self-organization
Angular forces
Biconnectivity

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 of a node toward one another when the corresponding angle is larger than 60° (even if these two nodes are not themselves 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 both types of forces, whose combination poses a number of technical challenges. We also provide an implementation that relies only on position exchanges within two hops. Extensive simulations are finally presented to evaluate the solution against all criteria (coverage, biconnectivity, quantity of movements, and diameter), and show its advantages over prior solutions.

1. 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 monitoring, collecting, or processing information sensed in the surrounding environment.

The efficiency and reliability of such a network 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 any of the nodes is removed. Besides fault-tolerance, it is important to *maximize the collective coverage* of the network (that is, the overall area the robots can sense 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 using as few *movements* as possible.

The problem of maximizing the coverage of a network of robots was addressed in numerous works, using either virtual repulsion forces (or a combination of repulsion and attraction forces, called *spring forces*) [9,23,8,11,22,20] or geometrical approaches [2,11,12,1] that equivalently regulate the inter-nodal distance and thus locally arrange the topology as a (equilateral) triangle tessellation, whose optimal properties with respect to coverage are well-known.[1] As a side effect, these approaches can establish biconnectivity at places where the density of robots is sufficiently high – but not in the general case.

Biconnectivity has been explicitely addressed in other works. In the most general context (targetting arbitrary, sparse, and even possibly disconnected topologies), solutions based on a common *point of interest* (*POI*) toward which all robots can converge and biconnect were proposed (e.g. see [8,18,15,14,13]). Having a common reference point known by all the robots simplifies the

☆ A preliminary version of this paper appeared in 72nd IEEE Vehicular Technology Conference (VTC'10-Fall).

* Corresponding author.
E-mail addresses: casteig@site.uottawa.ca (A. Casteigts), albert@labri.fr (J. Albert), chaumett@labri.fr (S. Chaumette), anayak@site.uottawa.ca (A. Nayak), ivan@site.uottawa.ca (I. Stojmenovic).

[1] The idea of using virtual forces to move individual robots seems to have first appeared in [3], then first used for a coordination problem in [17].

0140-3664/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2011.09.008

biconnectivity problem substantially (at least, without obstacles); this assumption is however not realistic in several practical scenarios. Different approaches were proposed in [4,7] to biconnect networks that are already connected based on movements of well chosen robots called *non-critical*; these solutions do not consider the coverage nor the diameter of the resulting network. In addition, the solutions in [4] are centralized, and the algorithm in [7] fails to biconnect in numerous situations (which is explicitly conceded by the authors). A similar approach by movement of so-called *removable* nodes was considered in [19] and [21] with the objective of minimizing movements by the nodes.

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 inter-connect (this feature can be found in any repulsion-based solution). Besides obtaining a much higher success rate than [7] (in the order of 95% against 50%), the main novelty of our solution 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 (see Fig. 1). This is done regardless of whether these neighbors are already in range of each other, so equilateral triangles are also formed at sparse places.

At a global scale, angular forces have the effects of biconnecting the network and reducing its diameter at the same time. The intuition of this behavior can be obtained by looking at Fig. 2.

The paper is organized as follows. The assumptions, notations, and network model are given in Section 2. Section 3 presents our solution and describe the technical challenges that had to be faced to successfully mix angular forces with spring forces (without negative interference between them). A possible implementation is then proposed in Section 4. Note that the principle of rotation can be implemented using at least two possible approaches, depending on which 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 thus briefly discuss some of the simulation results we obtained in Section 5, and eventually conclude with some remarks.

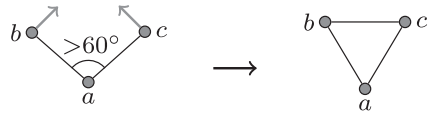


Fig. 1. Angular force principle. Here the angle formed by b and c relatively to a makes them rotate toward one another (around a).

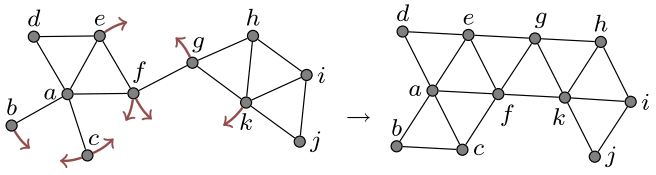


Fig. 2. Intuitive example of the effect of angular forces.

2. 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 (although we believe the proposed solution could be transposed in a framework with *relative* positions only). Each robot has *communication* and *sensing* ranges that are both 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 extra data within the beacon) and use this capability to share two-hops coordinate information among neighbors (up to 6 nodes positions in a given beacon). No additional communication is required. Finally, as the robots apply the algorithm, they regulate their inter-distance around a threshold d_{th}, which may be chosen by policy depending on the type of coverage desired (*e.g.* focuse or non-focused, see Fig. 3). The only constraint is that $d_{th} \leqslant\sim 0.851 C_R$ (for physical reasons that are precisely explained in Section 3.5), with the consequence that a slightly higher communication range will be required compared to solutions based on a common reference point (*e.g.*, for focused coverage, this leads $C_R \geqslant 2.035 S_R$ instead of the usual $C_R \geqslant \sqrt{3} S_R$ ($\simeq 1.732 S_R$).

3. The proposed virtual force scheme

3.1. 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). The downside of this approach is that such force vector, if mapped into a *discrete* movement of the considered node, may lead this node to move further than the equilibrium location, thereby generating oscillations. In general terms, these vectors loose track of the exact target location.

For this reason – and also because it simplifies the combination of different types of forces, we represent individual forces as *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, and keep track of the equilibrium point through direction and magnitude. As for the effective strength of a force, it is determined by the *product* of magnitude by weight.

(a) Focused coverage (b) Non-focused coverage

Fig. 3. Focused and non-focused coverages. *Circles* represent individual sensing coverages, *lines* represent communication links.

Please cite this article in press as: A. Casteigts et al., Biconnecting a network of mobile robots using virtual angular forces, Comput. Commun. (2011), doi:10.1016/j.comcom.2011.09.008

3.2. 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 forces involves a pair of node (whose *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). The cases in which each type of force apply, or not, is also discussed later.

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 that will be defined later, 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.

Angular forces

For a given angle $\gamma = \widehat{abc}$, where a and c are two consecutive neighbors of b in the clockwise direction (see Fig. 4 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 $\gamma \leqslant 60^\circ$, 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.*, $\pm 90^\circ$ relative to the angle of the segment from the node to the center, depending on their relative position ($\theta_{ab} + 90^\circ$ for a and $\theta_{cb} - 90^\circ$ for c, in the example of Fig. 4).

Remark 1. 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 4).

3.3. Cases where forces apply

Both spring and angular forces do not systematically apply among all neighbors. We consider to types of restrictions: *neighbor selection* (for both spring and angular forces), which causes a node to interact only with a well-chosen subset of direct neighbors, ignoring completely those neighbors that are said to be *shielded* by others; and *angle selection* (for angular forces only), whose effect is to ignore angular forces generated by the largest local angle when less than 6 neighbors are selected.

Neighbors selection (both types of forces)

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 $60^\circ \times 6 = 360^\circ$). The strategy proposed in [8] is to have some neighbors *shielding* others (*i.e.,* making them locally ignored). 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 a threshold of 60°. This scheme is illustrated on Fig. 5.

We adopt the same principle but consider a threshold of 54° instead of 60°. This precise value stems from the combination of two observations and corresponds to the best tradeoff in their respects. Let us first observe that a threshold of 60° does not allow to select more than 5 neighbors in practice (unless the corresponding 6 angles are *exactly* 60°). The right value to consider for a shielding angle – as far as triangle tessellation is targeted – should instead be of $360/7 + \epsilon$ (*i.e.*, $\simeq 51.43^\circ$), that is, the value that maximizes the chance to select 6 neighbors, while still preventing the selection of 7.

However, such a small value poses a different problem for the application of spring forces: it causes attraction to maintain pentagonal configurations as if they were stable structures, which is not desirable. In a pentagonal configuration (see *e.g.* Fig. 6), each sub-triangle has inner angle $360/5$ and outer angles $(180 - (360/5))/2 = 54^\circ$). Raising the shielding angle to at least 54° has the convenient effect that at least one sub-triangle of a pentagon will be broken as a by-product of shielding (*here, for exemple, b shieds c at a and a at c, which allows the edge ac to be discarded*).

Therefore, in order to maximize the selection of 6 neighbors, while preventing the stabilization of pentagonal configurations, we consider a shielding angle of 54° precisely. This is a crucial aspect of the solution.

Angle selection (for angular forces only)

Given a node and x angles formed by its x selected neighbors, angular forces do not apply relatively to the largest angle if $x < 6$. This restriction on angular forces is consistent with that of

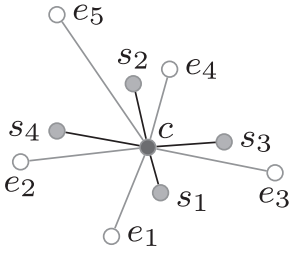


Fig. 5. Example of neighbor selection. *(Here nodes s_1, s_2, s_3, and s_4 shield nodes e_1, e_2, e_3, e_4, and e_5).*

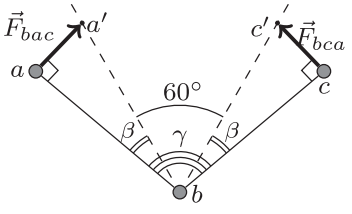


Fig. 4. Rotation with respect to b.

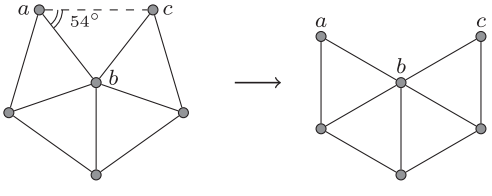


Fig. 6. Minimal shielding angle to prevent the maintenance of pentagonal formations.

Please cite this article in press as: A. Casteigts et al., Biconnecting a network of mobile robots using virtual angular forces, Comput. Commun. (2011), doi:10.1016/j.comcom.2011.09.008

neighbor selection seen previously. Considering again the example on Fig. 6, this will cause angular forces not to oppose the opening of angle $\widehat{abc}$ to the satisfactory extent. Another example is angle $\widehat{bad}$ in Fig. 2.

3.4. 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 thus 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 is 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 acting 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*.

3.5. 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 Fig. 2).

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 Fig. 7).

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 Fig. 8. Hence, the strength of the angular forces (that is, the product *magnitude* × *weight*) must increase as the correction decreases, still being equal to 0 when the correction is 0 (angle of 60°), and finally

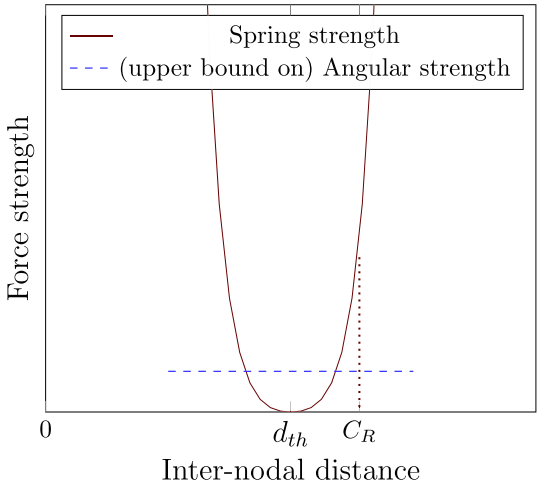


Fig. 7. Priority between forces. Angular forces must prevail only when spring forces are nearly satisfied. An additional constraint is to ensure that the angular strength is always significantly lower than the spring strength at $d_{th}/0.851$ (minimal C_R), so that angular forces cannot disconnect the network in case of conflicting configuration.

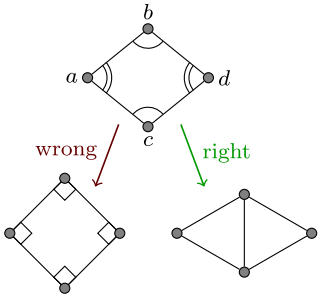


Fig. 8. Example of competition between two configurations. If the angular strength was designed to *increase* with the correction angle, this example configuration would become a square and stabilize as such, which is not a desired outcome. Now, if the strength *decreases* with the correction angle, 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).

be continuous over all its definition domain, from 0 to $\pi/3$ (the correction at 180° is $\pi/3$), which generates an apparent contradiction.

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 Fig. 9, 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 Fig. 9), 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 $d_{th} < 0.851C_R$.

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 Fig. 10). 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* × *weight*) is thus equal to the product

Please cite this article in press as: A. Casteigts et al., Biconnecting a network of mobile robots using virtual angular forces, Comput. Commun. (2011), doi:10.1016/j.comcom.2011.09.008

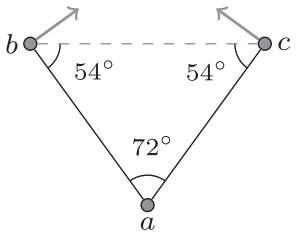


Fig. 9. Configuration of maximal strength.

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

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, then finding what value of a leads to a zero in the derivative at $\pi/30$. The derivative of the strength is

$$\begin{aligned} strength'(\beta) &= (tan'(\beta) \times e^{-a\beta^b}) + (tan(\beta) \times e^{t^{-a\beta^b}}) \\ &= e^{-a\sqrt{\beta}}\left(tan^2(\beta) - \frac{a \times tan(\beta)}{2\sqrt{\beta}} + 1\right). \end{aligned} \quad (2)$$

Replacing β with $\pi/30$, this derivative comes to zero when

$$a = \frac{\sqrt{30\pi} \times (tan^2(\frac{\pi}{30}) + 1)}{15 \times tan(\frac{\pi}{30})} \simeq 6.222 \quad (3)$$

which gives a (preliminary) weight of $e^{-6.222\sqrt{\beta}}$. The shape of this weight is illustrated on Fig. 10. Note that the ratio between $strength(\pi/30)$ and $strength(\pi/3)$ is now ~ 4.73.

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 3.3), we limit here the multiplicative factor so

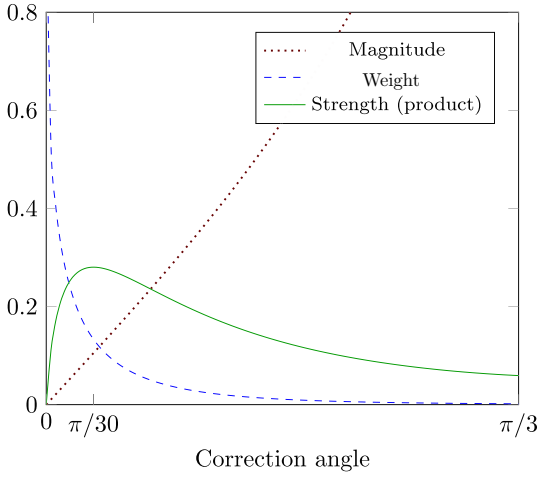


Fig. 10. Strength of angular forces (*the plot of the strength is increased 20 times for visibility purpose*).

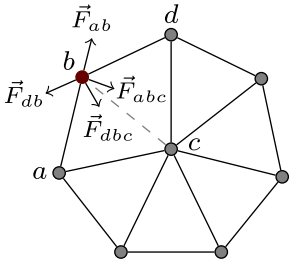


Fig. 11. Breaking heptagonal configurations. Due to the neighbors selection scheme, at least one node, b on the figure, will be discarded by c, and will therefore not interact with it. The desirable behavior here would be that the spring repulsion from a and d ($\vec{F}_{ab}$ and $\vec{F}_{db}$) push b away in the direction $\theta_{\vec{cb}}$. 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.

as to prevent *heptagonal* (or higher order polygonal) configurations (as explained in Fig. 11).

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 happened at a factor $\sim$1.15. The final formula for the weight of angular forces is thus

$$weight_A(\beta) = 1.15 \times e^{-6.222\sqrt{\beta}} \quad (4)$$

where β is the angular correction in radians. Finally, we can notice that the resulting strength satisfies the constraint mentioned in Fig. 7 (angular forces cannot prevail over spring attraction to disconnect the network). In fact, we precisely have $strength_S(correction(d_{th}/0.851)) \simeq 5.92 \times strength_A(\pi/30)$.

3.6. 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 and the other metrics (biconnectivity, diameter, coverage), as discussed in Section 5.6.

4. Implementation

We propose an implementation based on the periodical exchange of beacon messages conveying two-hops position information. More precisely, each beacon contains the (future) position of its emitter, along with the (expected) position of its emitter's selected neighbors. Upon reception by a robot, the beacons are stored in its local *mailbox*, which is then read offline at regular intervals (rounds). We assume a common round duration t_{rnd} for all robots (which can be for example inferred from the robots maximal speed, or specified before deployment).

Informally, the algorithm is as follows. In each round, robots starts by reading all the messages that arrived during the previous round in their mailbox, deduce their new list of neighbors and update (or create) local variables to store the corresponding positions (the positions of these neighbors and the positions of these 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

Please cite this article in press as: A. Casteigts et al., Biconnecting a network of mobile robots using virtual angular forces, Comput. Commun. (2011), doi:10.1016/j.comcom.2011.09.008

toward the next position. The detailed process is given in Algorithm 1.

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

```
1: neighbors[] ← ∅
2: messages[] ← mailbox.getMessages()
3: forall msg ∈ messages[] do
4:    neighbors[] ← neighbors[] ∪ msg.sender
5:    neighbors[msg.sender].position ← msg.pos
6:    neighbors[msg.sender].selection[] ← msg.selection[]
7: endfor
8: me.selection[] ← select(me, neighbors[])
9: nextP ← computeNextPosition(me)
10: send(sender = me, pos = nextP, selection = me.selection[])
11: if nextP ≠ currentPos then
12:    moveTo(nextP)
13: endif
```

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. $computeNextPosition(me)$

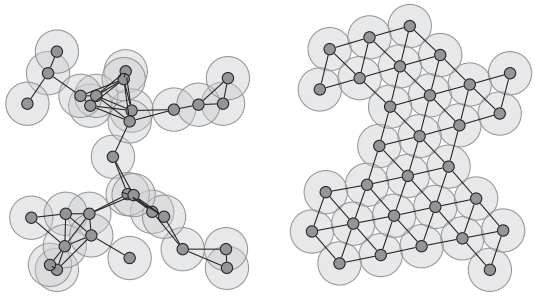
```
1: List[] ← ∅
2: forall ng ∈ me.selection[] do
3:    if me ∈ ng.selection[] then
4:       List[] ← List[] ∪ getSpringForce(ng, me)
5:       List[] ← List[] ∪ getAngularForce(ng, me)
6:    endif
7: endfor
8: V_res ← getWeightedAverage(List)
9: d = min(|V_res| × d_th, d_max)
10: if d ⩽ epsilon then
11:    d ← 0
12: endif
13: V_res ← (θ_{V_res}, d) // V_res is truncated to a magnitude of d
    (polar notation)
14: return current_position + V_res
```

In the context of our simulations, we assumed that robots can move at some speed v_{max} with instantaneous acceleration, instantaneous direction change, and negligible computation time, that is, up to a distance of $d_{max} = v_{max} \times t_{rnd}$ 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, since robots do not substantially change their direction from one iteration to the next (which we believe is due to a better appreciation of their target by using two-hop information and averaging instead of summing).

5. Experimentations

5.1. Summary of the results

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



(a) Initial topology (b) Resulting topology

Fig. 12. Example of arbitrarily connected topology of 30 nodes (and corresponding outcome with our algorithm).

- 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 Fig. 12).

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 the distributed algorithm from [7], the only known distributed solution addressing such scenarios. Finally, we evaluated our algorithm further in Case *C* by studying the impact of *epsilon* (the threshold below which robots do not move at all) on the four metrics. The complete set of simulations results is given in the next section.

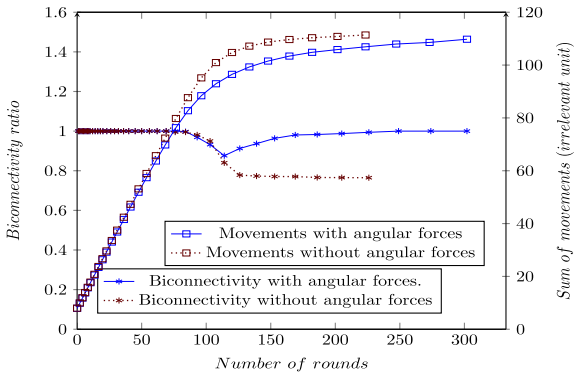
In a nutshell, the combination of both kinds of force (our solution) prove relevant in Case A (using slightly less movements than spring forces alone, and maintaining the biconnectivity of the whole group whereas spring forces alone do not; however the combination induced a cost of ~25% more time to stabilize). In Case B, our solution prove *not* relevant, and even detrimental due to its requirement of having $d_{th} \leqslant 0.851 C_R$ (this requirement can be released when using spring forces only). In Case *C*, where the topologies were generated by drawing the positions of nodes uniformly at random (with an appropriate density [5]), then selecting only the connected ones for simulations, our algorithm achieved biconnectivity in more than 90%, against 50% for the algorithm in [7]. It also led to 60% more coverage, and 8% less diameter (although this excellent coverage is specific to our solution and can be observed in any algorithm having a *repulsion*-based mechanism). Finally, by studying the impact of *epsilon*, we clearly highlighted how this parameter can be used to leverage the priority among the metrics, in particular the tradeoff between *movements* and *biconnectivity*. All simulations were performed using the *JBotSim* platform [6].[2]

5.2. Details of the simulation results

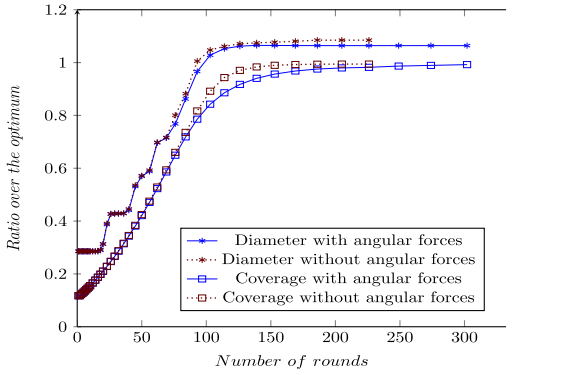
In all simulations but one explicitly mentioned, we considered a non-focused coverage with $d_{th} = 2S_R$. As for C_R, we used the smallest value allowed by the solution (that is, $\frac{d_{th}}{0.851} \simeq 2.35 S_R$).

[2] An interactive demo. of the algorithm is available at: <http://jbotsim.sf.net/examples/>

Please cite this article in press as: A. Casteigts et al., Biconnecting a network of mobile robots using virtual angular forces, Comput. Commun. (2011), doi:10.1016/j.comcom.2011.09.008



(a) Biconnectivity and movements.



(b) Diameter and coverage

Fig. 13. Combination of spring and angular forces *vs.* spring forces alone.

5.3. Case A

In Case A, nodes are released as a (random) high-density conglomerate. The purpose here is to determine whether the combination of angular forces and spring forces behave better than spring forces alone in this context (where the initial topology is already biconnected). Put differently, we answer the question of whether the 'angular component' of our algorithm should be preferably enabled or disabled in this context. The test involved 40 robots randomly distributed within a very small area (of size C_R^2). For each so-generated topology, we ran the algorithm with and without angular forces, and measured the biconnectivity, coverage, diameter, and quantity of movements made at the end of the execution. The results, averaged over 200 simulations (a different topology for each), are shown in Fig. 13. Note that the diameter and the coverage (Fig. 13) are expressed as a ratio over the optimum, that is, a coverage of 1 corresponds to non-overlapping individual coverages, and a diameter of 1 corresponds to the smallest diameter supposedly achievable (by setting up a concentrically layered triangular tessellation of inter-nodal distance d_{th}).

Interpretation. A first observation is that the algorithm terminates faster without angular forces (25% less rounds needed in average), which makes sense knowing that angular forces start to efficiently manifest after spring forces have approached the equilibrium. Still, the final amount of movements remains lower when angular forces are used, possibly due to a finer appreciation of the moving trajectory by considering two-hops information. The main result here is that using angular forces led to a biconnected topology in every case, whereas spring forces alone did not. In fact, at some point (after ~100 rounds), both algorithms lost biconnectivity (due to the neighbor shielding mechanism which cancels some attraction forces and allows border nodes to be pushed further away). In the case of angular forces, the biconnectivity is subsequently restored. Regarding the coverage and diameter, both algorithm perform rather comparably. Both eventually reach an optimal coverage (although faster without angular forces), and similar diameters (with a small advantage for angular forces). The conclusion is that angular forces are relevant in Case A, and particularly if biconnectivity is a important criterion.

5.4. Case B

In the case of a closed area, we mentioned that repulsion alone could be used to inter-connect isolated arbitrarily distributed robots. For a given square area and a given communication range, we are thus interested in studying how the density of nodes influences the way nodes connect to each other during the algorithm execution. More precisely, we tried to evaluate the impact of our assumption that $d_{th} \leqslant 0.851C_R$, by comparing it to spring forces for which we set $d_{th} = 0.98C_R$ (the 2% margin is to prevent nodes from pushing each other out of range). The impact of walls was implemented as follows: each node normally computes its resulting vector according to its neighbors. Then, if the resulting vector points to a location closer to the wall than the desired value, it simply subtracts the difference as an orthogonal component from the wall (as illustrated on Fig. 14), then move as per the so-obtained vector. Note that this implementation requires that the node is able to detect a wall from farther than the forbidden area.

We generally recommend to implement obstacles using similar approaches, rather than using, e.g., a virtual repulsion force of infinite weight – which would dilute the impact of neighboring nodes.

The experiment involved a square area roughly corresponding to a capacity of 60 robots non-overlapping in coverage. The number of randomly deployed robots in this area was varied from 10 to 55 and we measured, for both algorithms, the number of connected components obtained at the end of the execution. The results, whose average over 100 topologies for each number of nodes is given Fig. 15, show that significantly more nodes are required to connect when angular forces are used (here, around 23% more). As a conclusion, we advise to *dismiss* the use of angular forces in Case B.

5.5. Case C - Comparison with the algorithm from [7]

The Case C corresponds to initial topologies that are arbitrarily distributed but connected. We know since [5] that the probability of connectedness of a set of nodes distributed at random (uniformly) exhibits a sharp transition from 0 to 1 around a given density threshold. Randomly connected topologies can thus correspond to practical situations in which robots are for example thrown over a given area in a *sufficient* number. We compared here our algorithm to the one from [7] (subsequently referred to as *DLNS*), which is the only known competitor in Case C.

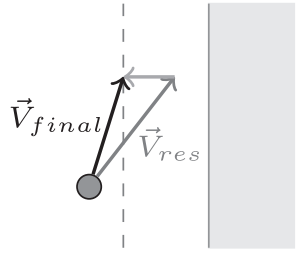


Fig. 14. Implementation of the walls. *The dashed line represents the minimal distance from the wall.*

Please cite this article in press as: A. Casteigts et al., Biconnecting a network of mobile robots using virtual angular forces, Comput. Commun. (2011), doi:10.1016/j.comcom.2011.09.008

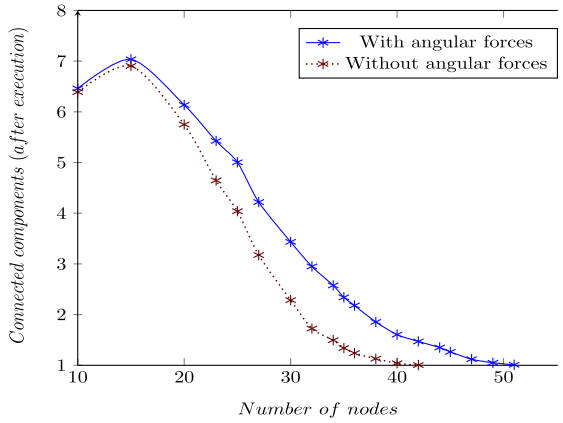


Fig. 15. Number of nodes required to connect in a closed area (Case B).

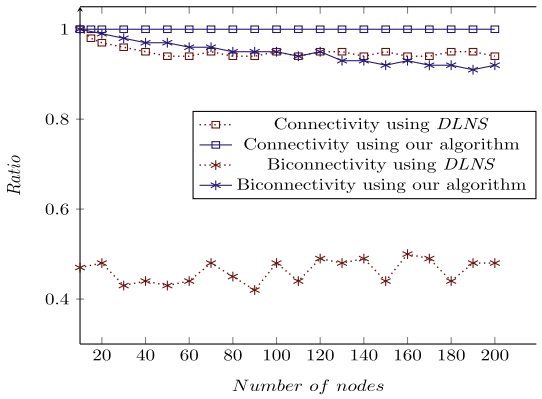


Fig. 16. Average connectivity and biconnectivity at the end of the execution (in function of the number of nodes).

We used the results from [5] to generate topologies that are connected, but still far from being biconnected. This was done by using the probability equation of the minimum degree given in that paper (transcripted on Eq. 5 below), which converges asymptotically (as n grows) to the same probability as the k-connectivity [16].

$$P(d_{min} \geqslant k) = \left(1 - \sum_{i=0}^{k-1} \frac{(\rho\pi C_R^2)^i}{i!} \cdot e^{-\rho\pi C_R^2}\right)^n \quad (5)$$

In this equation, n is the number of nodes, and ρ is the density. In order to generate connected topologies that are far from being biconnected, we set empirically the density ρ so as to observe $\frac{P(d_{min}\geqslant 1)}{P(d_{min}\geqslant 2)} \simeq 100$, then among the topologies generated using this density, we kept only those being indeed connected, and not biconnected.

An example of such topology is depicted on Fig. 12. The behavior of both algorithms was compared upon the four metrics. The averaged results are given on Fig. 16 (for connectivity and biconnectivity) and Fig. 17 (for coverage, diameter, and movements).

Interpretation. First of all, let us stress that our algorithm *never* disconnects the network. Biconnectivity is successfully achieved in more than 90% of the cases (more than 95% for less than 100 nodes). *DLNS*, on the other hand, biconnects in approximately 50% of the cases, and sometimes disconnects the network. The results concerning the connectivity and biconnectivity are thus clear-cut in the advantage of angular forces.

Regarding the other metrics, *DLNS* has a neat advantage for movements, and is more scalable in this respect, since the average movements per robot with our algorithm grows regularly with the number of nodes whereas is becomes asymptotically constant with *DNLS*. The same trend was observed for the number of rounds before stabilization, which was about twice higher for 200 nodes (in the order of 2000 rounds) than for 20 nodes ($\sim$ 1000 rounds) in our case, whereas the number of rounds used in average by DLNS remained independent from the number of nodes (60 rounds in average). The very high number of round of our solution is a common trait among virtual forces based algorithms, in which nodes perform small and incremental movements. This may be seen or not as a problem, depending on whether the energy consumed by beacons is neglected in comparison to physical movements – these aspects are beyond the scope of the paper.

Finally, the coverage offered by our solution is higher by 60% (as already discussed, the coverage with virtual forces is in fact optimum); and the resulting diameter is smaller by $\sim$8% for our solution.

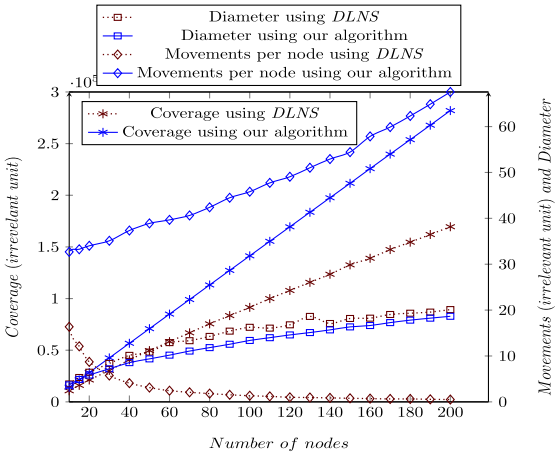


Fig. 17. Average coverage, diameter, and movements in function of the number of nodes (when biconnectivity is successfully achieved).

5.6. Case C - Independent benchmark

In this last set of experimentation, we study our virtual force algorithm further in the Case *C* (arbitrarily connected topologies). The purpose is to study the impact of the threshold *epsilon* on the algorithm outcome. For each execution, we measured the four metrics at the end of the execution. The results are shown on Fig. 18, using a logarithmic scale for epsilon. Note that *all* the metrics are here given as a ratio over their optima, including for the movement metric: we compared here the overall (that is, collective sum of) movements performed to the smallest amount of movements that could have generated the same topological outcome. This is done as follows: for a given initial topology T_i and a final topology T_f, we shift T_f so as to align its barycenter with that of T_i, then compute the *minimum weighted bipartite matching* between both topologies, using the heuristic from [10] with edge lengths as the weights. The so-obtained sum of weights is considered as the optimal amount of movements.

Interpretation. The parameter *epsilon* has a tremendous impact on the outcome. A small value clearly benefits biconnectivity at the expense of more movements, and reciprocally. Besides, coverage and diameters are both better for smaller values, although they do not degrade rapidly with the growth of *epsilon*. The conclusion is that *epsilon* can be used to leverage the trade-off between biconnectivity and movements. A more detailed expression of this trade-off, *e.g.* as a balance equation between the four metrics, is planned for future work.

Please cite this article in press as: A. Casteigts et al., Biconnecting a network of mobile robots using virtual angular forces, Comput. Commun. (2011), doi:10.1016/j.comcom.2011.09.008

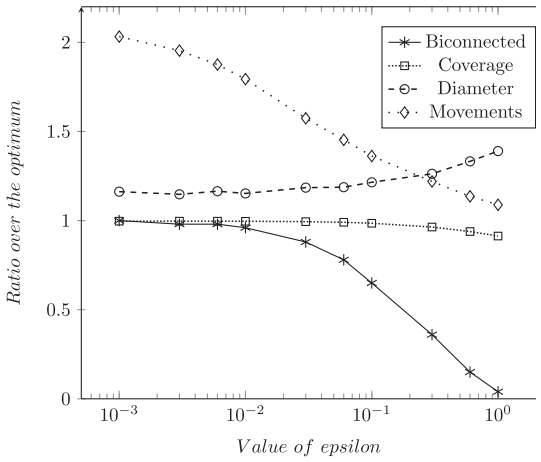


Fig. 18. Average biconnectivity, coverage, diameter, and movements over their optima, in function of the threshold *epsilon* (for 50 nodes).

6. Concluding remarks

The use of virtual forces in general, and spring forces in particular, to self-deploy swarms of autonomous robots has been advocated the past few years for their elegance and good properties. Beyond being purely localized, virtual forces allow to maximize the coverage using repulsion, while maintaining the connectivity using attraction. In this paper we investigated the joint use of such forces with a new kind of forces called *angular* forces, which have the effect of *contracting* large angles formed by two consecutive neighbors at a same node. 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 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 in the nodes movements.

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 cases (i). A related question of interest is whether enabling or disabling angular forces could be decided adaptively by the robots themselves, or if this should be set *a priori* (by anticipating the context of use). Regarding the 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 (and at a lesser extent the coverage and the diameter obtained). It would be interesting to characterize this trade-off further in a future work, possibly providing an equation that makes it possible to balance the four metrics at once and thus allowing the algorithm to adapt easily to several contexts and requirements.

Acknowledgments

This work was partially supported by NSERC CRDPJ 386874-09 (Reliable and secure QoS routing and transport protocols for mobile ad hoc networks), NSERC STPSC 356913-2007 (Maintaining fault-tolerant networks of robots for supporting wireless sensor networks), by grant Innovative electronic components and systems based on inorganic and organic technologies embedded in consumer goods and products, TR32016, Serbian Ministry of Science and Education, by the French DGCIS (Direction Générale de la Compétitivité, de l'Industrie et des Services), and by the ITEA 2 Smart Urban Spaces project consortium.

References

[1] K. Akkaya, I. Guneydas, A. Bicak, Autonomous actor positioning in wireless sensor and actor networks using stable-matching, Int. J. Parallel Emergent Distrib. Syst. 25 (6) (2010) 439–464.
[2] K. Akkaya, 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), New Orleans, Louisiana, (2007) 281–288.
[3] Ronald C. Arkin, Motor schema–based mobile robot navigation, Int. J. Robot. Res. 8 (4) (1989) 92–112.
[4] P. Basu, J. Redi, Movement control algorithms for realization of fault-tolerant ad hoc robot networks, IEEE Network 18 (4) (2004) 36–44.
[5] 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 AdHoc Networking and Computing, ACM, New York, NY, USA, (2002) 80–91.
[6] A. Casteigts, The JBotSim Library, CoRR, abs/1001.1435, (2010).
[7] S. Das, H. Liu, A. Nayak, I. Stojmenovic, A localized algorithm for bi-connectivity of connected mobile robots, Telecommun. Syst. 40 (2009) 129–140.
[8] M. Garetto, M. Gribaudo, C.-F. Chiasserini, E. Leonardi, A distributed sensor relocation scheme for environmental control, IEEE Intl. Conf. on Mobile Adhoc and Sensor Systems (MASS'07), (2007) 1–10.
[9] A. Howard, M. Mataric, 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.
[10] Harold W. Kuhn, The hungarian method for the assignment problem, Nav. Res. Logist. Q. 2 (1955) 83–97.
[11] G. Lee, N.Y. Chong, A geometric approach to deploying robot swarms, Ann. Math. Artif. Intell. 52 (2–4) (2008) 257–280.
[12] M. Li, J. Harris, M. Chen, S. Mao, Y. Xiao, W. Read, B. Prabhakaran, Architecture and protocol design for a pervasive robot swarm communication networks, Wireless Comm. and Mobile Comp. (2009).
[13] X. Li, R. Falcon, A. Nayak, I. Stojmenovic, Servicing Wireless Sensor Networks by Mobile Robots, IEEE Commun. Mag. (2011), in press.
[14] X. Li, H. Frey, N. Santoro, I. Stojmenovic, Focused Coverage by Mobile Sensor Networks, IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'09), Oct. 2009.
[15] H. Liu, X. Chu, Y.-W. Leung, R. Du, Simple movement control algorithm for bi-connectivity in robotic sensor networks, IEEE J. Sel. Area Commun. 28 (7) (2010) 994–1005.
[16] M.D. Penrose, On k-connectivity for a geometric random graph, Random Struct. Algorithms 15 (2) (1999) 145–164.
[17] F.E. Schneider, D. Wildermuth, H.L. Wolf, Motion coordination in formations of multiple mobile robots using a potential field approach, Distrib. Auton. Robot. Syst. 4 (2000) 305–314.
[18] G. Tan, S.A. Jarvis, A.-M. Kermarrec, Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks, in: 28th International Conference on Distributed Computing Systems (ICDCS'08), June 2008, 429–437.
[19] Z. Yan, Y. Chang, H. Jiang, Z. Shen, Fault-tolerance in wireless ad hoc networks: Bi-connectivity through movement of removable nodes, Wireless Commun. Mobile Comp. in press.
[20] S. Yoon, O. Soysal, M. Demirbas, C. Qiao, Coordinated locomotion and monitoring using autonomous mobile sensor nodes, IEEE Trans. Parallel Distrib. Syst. 22 (10) (2011).
[21] S.G. Wang, X.F. Mao, S.J. Tang, X.Y. Li, J.Z. Zhao, G.J. Dai, On movement-assisted connectivity restoration in wireless sensor and actor networks, IEEE Trans. Parallel Distrib. Syst. 22 (4) (2011) 687–694.
[22] M. Xi, Y. Qi, K. Wu, J. Zhao, M. Li, Using potential to guide mobile nodes in wireless sensor networks, Adhoc Sensor Wireless Networks 12 (3–4) (2011) 229–251.
[23] Y. Zou, 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), March 2003, vol. 2, 1293–1303.

Please cite this article in press as: A. Casteigts et al., Biconnecting a network of mobile robots using virtual angular forces, Comput. Commun. (2011), doi:10.1016/j.comcom.2011.09.008