An Ant-Based Algorithm for Annular Sorting

Martyn Amos and Oliver Don

***Abstract*— We present an ant-based algorithm for spatially sorting objects into an annular structure. The model is minimal, as it requires only stochastic ant behaviour coupled with a pheromone-inspired "attraction-repulsion" mechanism. This is the first annular sorting algorithm to consider the problem of objects with non-uniform size, as well as the situation where objects are pre-sorted. The algorithm consistently generates high-quality annular structures, and is particularly powerful in situations where the initial configuration of objects is similar to those observed in nature. We suggest that this model provides a powerful algorithmic framework, as well as a contribution to the study of "attraction-repulsion" models of animal behaviour.**

I. Introduction

The ability of social insects to collectively solve problems has been well-studied and documented [1]. The behaviour of foraging ants, for example, has been abstracted to provide algorithmic solutions that are robust, distributed, and flexible [2], [3]. The particular behaviour that we will focus on is the clustering or sorting of ant corpses or larvae [4]. Abstract models of these behaviours have been successfully applied to, amongst other problems, numerical data analysis, data mining, and graph partitioning [5]. In this paper, we focus on the task of *brood sorting*. This behaviour, when observed in *Temnothorax unifasciatus*[1] [6], leads to the formation of a single cluster of offspring made up of concentric rings of brood items, with the youngest items (eggs and micro-larvae) being tightly packed at the centre, and successively larger larvae arranged in increasingly wider-spaced bands moving out from the centre of the cluster. Models of this behaviour have yielded new algorithmic solutions to the problem of annular sorting [7], [8], [9], [10], [11].

In this paper we present a novel algorithm inspired by an intriguing hypothesis by Franks *et al.* concerning the biological mechanisms underlying annular sorting. In their article, the authors state that "The mechanism that the ants use to re-create these brood patterns when they move to a new nest is not fully known. Part of the mechanism may involve conditional probabilities of picking up and putting down each item which depend on each item's neighbours ... The mechanisms that set the distance to an item's neighbour are unknown. They may be pheromones that the brood produce and which tend to diffuse over rather predictable distances ..." [6]

We should note that the purpose of this paper is *not* to investigate the actual biological phenomenon in question; we simply use it as inspiration for developing a new algorithmic technique. We have constructed a corresponding minimal model, using only stochastic ant behaviour and pheromone-style "repellents" and "attractants"; this model gives rise to the emergence of annular clusters of objects in simulated *Temnothorax* colonies. Our algorithm is competitive with existing solutions for simple sorting, and has the additional properties of being able to deal with both objects of non-uniform physical size and pre-sorted piles of items.

In Section II we present the background to the problem, before describing our model in Section III. In Section IV we present and discuss the results of experimental investigations, and conclude in Section V with a discussion of their implications.

II. Annular Sorting

Franks *et al.* [6] carried out an observational biological study of the brood sorting behaviour of *Temnothorax* ants. The nesting behaviour of this species made them particularly suitable for study as they nest in single clusters in flat rock crevices, a situation that is easy to replicate and monitor in a laboratory. Photographs were taken of the ants' brood cluster and individual items were classified, before a tessellation was applied to determine density and distance from the cluster centroid. This study showed that *Temnothorax* ants placed smaller brood items at the centre of the cluster with a greater density, forming an annular arrangement. This process reasserted itself when the ants were forced to migrate to a new nesting site, and proved to be ongoing. This structure is illustrated in Figure 1, where three different types of brood item are arranged in roughly-sorted concentric rings.

Wilson *et al.* proposed the first model of "ant-like annular sorting" to simulate the behaviour of *Temnothorax* ants using minimalist robot and computer simulations [7]. Three models for annular sorting were presented: "Object clustering using objects of different size", "Extended differential pullback" and "leaky integrator". The first was run exclusively as a computer simulation, since modifying robots to allow them to move objects of different sizes proved to be too complex. Despite this, the computer simulation modelled physical robot behaviour faithfully, preserving the limitations of movement inherent in simple robots, and even going so far as to build in a 1% sensor error that matched the rate seen in the machines.

The first model explored the theory that annular sorting occurs solely due to the different sizes of the objects involved, Wilson *et al.* compared this to the manner in which muesli settles in transit, with smaller objects falling to the bottom, leaving the larger ones on top. The simulation modelled

Martyn Amos is with the Department of Computing and Mathematics, Manchester Metropolitan University, UK (email: M.R.Amos@mmu.ac.uk). Oliver Don contributed to this work whilst with the School of Engineering, Computer Science and Mathematics, University of Exeter, UK.

[1]Until recently, this species was known as *Leptothorax unifasciatus*.

1-4244-1340-0/07/$25.00

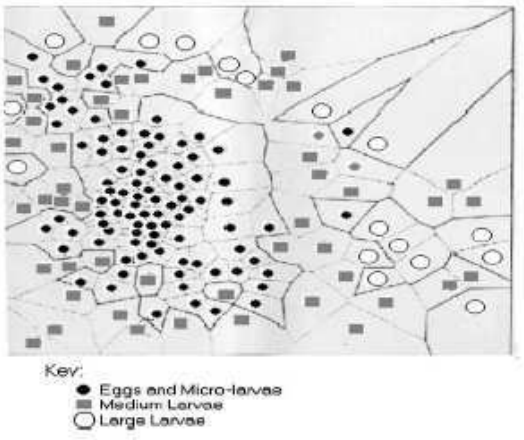

Fig. 1. Annular sorting in a real *Temnothorax* colony (taken from [6], with permission).

agents who picked up the first object they encountered while unladen and deposited it at the next object they encountered. The results displayed a slight increase in the quality of clusters when there was a greater number of different object sizes; however, clusters tended to form at the edge of the area and were often "inside out", i.e., with larger objects at the centre surrounded by smaller objects. This lead Wilson *et al.* to conclude that merely using objects of different sizes did "not create a sufficient muesli effect".

The second and third models attempted to recreate ant brood clustering behaviour by assigning more complex behaviour to the ants. Wilson *et al.* hypothesised that ants were able to recognise inherent differences in larval growth, and created annular clusters by depositing different objects at different differences away from each other depending on size. Due to the limitations of the robots, this was implemented by having the agents reverse a distance depending on the kind of object carried before depositing it. Initial results with this approach were not good. As a result Wilson *et al.* proposed a third model, calling it the "leaky integrator". This allowed the agents to have an adaptive amount of "pullback" that varied according to how many objects of the same kind had been encountered in the last n seconds. Initial tests with this system produced poor results but results improved when a genetic algorithm was used to select parameter values. Two subsequent models, due to Hartmann [8] and Vik [10], both use a neural network controller for individual ants, with network weights being evolved using a genetic algorithm. These models have been successfully applied to the problems of clustering and annular sorting of objects (with spatial restrictions imposed, see the later discussion.) Other related work has studied emergent sorting using cellular automata [11].

III. Our Proposed Model

We now propose an alternative algorithm for annular sorting. In contrast to previous work, we focus our attention on the items to be sorted rather than on the agents performing the sorting. Our algorithm is a distributed system in which "ants" probabilistically pick up or drop items depending on an assessment of the item's "score" (calculated as a function of its current position). Ants are represented as "agents", and brood items of different sizes are represented by "objects". Agents and objects are spatially distributed at random on a two-dimensional "board" of fixed size.

Each object has a *placement score*; agents move randomly across the board, and when they collide with an object they calculate its placement score. This score is then used to probabilistically determine whether the agent should *pick up* the object and become laden. Laden agents carry objects around the board, and at every time-step they evaluate what placement score the carried object *would have* if it were to be deposited at the current point. This score is then used to probabilistically determine whether the object should be *deposited*.

Each category of object to be sorted posses three attributes: *size, minimum perimeter* and *maximum perimeter*. Size is important, as objects may not overlap; the other two attributes are both functions of the object's size. Minimum and maximum perimeter are used in the calculation of the object's placement score: When evaluating the placement score of an object, the agent counts how many nearby objects fall within the minimum perimeter (these count towards a weighted penalty), and how many fall within the maximum perimeter (these count towards a weighted bonus).

It is important to note that the maximum and minimum perimeters are only calculated for the object whose placement score is currently being evaluated. Agents take no account of whether a placement results in a good or bad score for neighbouring objects, thus small objects with smaller minimum perimeters will frequently be placed quite close to larger objects, resulting in a penalty score for those larger objects. As a result groups of smaller objects will tend to "force out" larger objects. It could be argued that this process implements a stronger version of the size-based muesli analogy proposed by Wilson *et al.* [7] As stated earlier, the minimum and maximum perimeter values are a function of an object's size (we describe the exact function shortly). The minimum perimeter of an object acts as a "repulsive" force, whereas the maximum perimeter acts as an "attractive" force. These coupled forces ensure that objects maintain a proportional "exclusion zone", around themselves, whilst ensuring the coherence of a single cluster around some centroid ("centre of gravity"). A precedent for this "attraction-repulsion" model of collective behaviour already exists and has been well-studied in the ecological domain [12], [13]. We may speculate that a plausible biological encoding of perimeter values may use the concentration of some pheromone deposited by ants on objects. However, we have not yet tested this theory, and it is important to note that

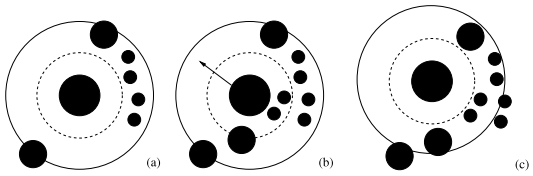

Fig. 2. Depiction of how large objects are forced out of clusters. (a) Good configuration. (b) Placement score of central object drops as a result of three new objects being placed within its minimum perimeter. (c) Object is replaced with a higher placement score.

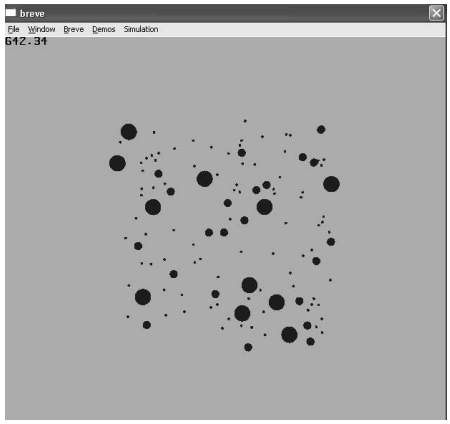

Fig. 3. Poor sorting caused by zero delay.

we are not claiming to have developed a pheromone-*based* algorithm.

We now illustrate the underlying principle of our model with an example, depicted in Figure 2. This example shows how larger objects are "forced" out of clusters by their proximity to other objects. The large object in Figure 2 has its minimum perimeter depicted as a dotted line, and its maximum perimeter as a solid line. In Figure 2(a), the object has a high-scoring placement score, as it has no objects within its minimum perimeter (which would attract a penalty), and several objects in its maximum perimeter (attracting a bonus score). However, if several objects are later deposited in the central object's immediate vicinity (Figure 2(b)), its placement score changes for the worse, as these objects contribute a significant penalty. As a result, the next time this object is encountered by an agent it will be carried until it once again has a beneficial placement score (Figure 2(c)), where it will be deposited. As we can see, this calculation of placement scores has the effect of moving larger objects away from clusters of smaller objects, whilst maintaining relative proximity.

Before embarking on a full implementation of our algorithm, we carried out test runs using a prototype system. This highlighted two, previously unforeseen, effects that we deal with in the full algorithm. Early investigations showed that ants had a tendency to construct good annular clusters, but would then deconstruct them at the outer edge, as there was no termination criterion defined for the algorithm. We solved this problem by introducing the notion of "energy"; each ant starts with a fixed amount of energy, represented as an integer value, which is decremented every time the ant picks up an object (the amount of energy lost is a function of the object's size). Once the agent's energy level reaches zero it is removed from the system, and the simulation terminates when there are no more agents left. This prevents the simulation from ending when objects are still being carried, but also has the beneficial effect of smoothly and gradually reducing the overall activity within the system as the simulation approaches termination. The biological validity of this approach is unclear, but it appears to make the placement of outer objects much more realistic.

Another problem that was highlighted by the prototype algorithm was that of "chaos"; ants would frequently deposit an object and then immediately pick it up again, as they were still in contact with it. Conversely, ants would also frequently deposit objects after taking a single "step", particularly if the placement score was moderately good. This behaviour creates clusters in which overall placement was quite poor, as large objects were rarely removed from the centre of the cluster (Figure 3). In order to deal with this problem, a "cooling down" delay period was introduced; agents must wait a set number of steps after colliding with or picking up an object before they may carry out any further placement calculations. This appeared to solve the problem; different (non-zero) values for the delay variable impacted only on the run-time of the simulation, and did not affect the quality of clusters generated. A delay value of 4 was chosen for what follows.

A. The Algorithm

We now describe in detail our algorithm for annular sorting. Ants are modelled by *agents*, each of which has the following attributes:

- Location (application representation)
- Laden (true/false)
- Object (application representation of object carried if Laden == true)
- Energy (integer)
- Delay (integer)

Objects are modelled as spheres, and have a single attribute, from which their minimum and maximum perimeters are calculated:

- Size (real from the set $\{0.5, 1.5, 3\}$, corresponding to small, medium or large)

The following constants are defined to deal with objects (all distances are measured from the edge of objects, rather than from their centre):

- BASESCORE (0)
- BONUS (0.1)
- PENALTY (-60)
- BONUSMULTIPLIER (4.0) (to calculate maximum perimeter)
- PENALTYMULTIPLIER (0.4) (to calculate minimum perimeter)

There are n objects and m ants initially distributed at random on a two-dimensional "board" of fixed size. The number of ants and numbers of objects of each size may be specified in advance. Ants may move over other ants or over objects; this is in contrast to previous work modelling robotic agents, where inherent spatial restrictions exist. We impose no such limitations, and discuss in a later section the implications for comparison of results. Movement may occur continuously in any direction on the Cartesian plane; we do not impose a discrete, cell-based "neighbourhood".
The pseudo-code expression of the algorithm is as follows:

```
while (agents exist)
    for all agents (A_1, ... A_i, ... A_m) do

        // Remove ant if "dead"
        if (A_i.Energy == 0)
            remove A_i from system
            break // i.e., go to next ant, if possible
        end if

        // Check delay
        if (A_i.Delay > 0)
            A_i.Delay = A_i.Delay -1
        else
            // Unladen ant collides with object
            if (A_i.Laden == false and
                (A_i.Location == some O_i.Location))

                A_i.Delay = 4
                Score = CalculatePlacement(O_i)
                Probability = random(0 ... 1)

                if (Score > Probability)
                    A_i.Object = O_i
                    A_i.Laden = true
                    A_i.Energy = A_i.Energy - O_i.Size
                end if
            end if

            // Laden ant in free space
            if (A_i.Laden == true and
                (A_i.Location == empty))
                Score = CalculatePlacement(A_i.Object)
                Probability = random(0 ... 1)
                if (Score > Probability)
                    Place A_i.Object at A_i.Location
                    A_i.Delay = 4
                    A_i.Laden = false
                end if
            end if

        Move A_i to a randomly selected adjacent location

    end for
end while
```

We now describe the CalculatePlacement() function:

```
CalculatePlacement(Object)

Score = BASESCORE

for each neighbouring Object N_i

    // "within" calculates Cartesian proximity
    if (N_i.Location within
        (PENALTYMULTIPLIER * Object.Size))
        Score = Score - PENALTY
    else
    if (N_i.Location within
        (BONUSMULTIPLIER * Object.Size))
        Score = Score + BONUS
    end if

end for

return (Score)
```

B. Implementation

The pseudo-code above was implemented using the Breve[2] multi-agent system environment [14]. This package facilitates the simulation of decentralised systems, in a similar fashion to the well-known Swarm[3] system. The benefits of using such a framework are derived from the fact that it automatically handles issues such as collision detection, construction of object neighbourhoods, and the user interface. This allows the programmer to concentrate effort on the important aspects of the model's implementation, rather than on "house keeping" issues. All simulations were run on an PC under Windows XP, although the code runs equally well on both Linux and Apple Macintosh machines. An additional Java application was written to deal with post-processing of object coordinates.

IV. Results and discussion

In order to assess the quality of sorted structures, we apply three performance metrics: *separation*, *shape*, and *radial displacement*, as defined in previous work [7]. Separation and shape are expressed as a percentage, with a value of 100% being interpreted as ideal. Separation measures the degree to which objects of similar size are kept apart from objects of differing size (i.e., the degree of "segregation"). The distance to the structure centroid is calculated for each object, and the upper and lower quartiles computed for each object type. We then perform three individual counts:

1) N_s = the number of small objects that have a distance to the centroid *greater* than the lower quartile range of either the medium or the large objects,

[2] Available at http://www.spiderland.org/breve/
[3] http://www.swarm.org

2) N_l = the number of large objects that have a distance to the centroid *less* than the upper quartile range of either the medium or the small objects,
3) N_m = the number of medium objects that have a distance to the centroid *greater* than the lower quartile range of the large objects, plus the number of medium objects that have a distance to the centroid *less* than the upper quartile range of the small objects. This count is then divided by two to prevent bias.

Separation is therefore expressed for n objects as

$$Se = 100 * (1 - \frac{N_s + N_l + N_m/2}{n}) \quad (1)$$

The shape metric is used to assess the "circularity" of the structure generated, with the ideal structure being a compact core of small objects, surrounded by perfectly circular successive bands of larger objects. This metric is calculated in two stages. We first calculate F_s, the fraction of small objects located in the central cluster. We achieve this by constructing a graph, with each vertex representing a small object, and an edge connecting two vertices if the corresponding objects are within 2.5 spatial units of one another. We then divide the size of the largest connected component of this graph by the total number of small objects. The second stage of the shape calculation involves finding the deviation from some common radius for each object size, since each object would ideally lie on the same radius as every other object of that size. For the medium and large objects, we first calculate the common radius by taking the mean radial distance from the centroid (r_m and r_l for medium and large objects). We then calculate the absolute deviation from this radius by summing the Euclidean distances to each object of a particular type from this radius (d_m and d_l for medium and large objects). We then calculate this as a percentage by placing this sum deviation between zero and one common radius. Shape is therefore expressed as (cluster fraction + sum of shape performances for medium and large objects)/number of object sizes:

$$Sh = (100F_s + (100 * (1 - (d_m/n_m * r_m))) + (100 * 1 - ((d_l/n_l * r_l))))/3 \quad (2)$$

Radial displacement is used to measure the "compactness" of a structure, and yields a distribution of distances from the centroid for each object type. Previous studies [7], [8] provide precise formulae for the calculation of compactness for a given structure, but this is difficult for our model. Earlier work used objects of uniform size, which makes the task of calculating an optimal "packing" relatively straightforward. Here, however, we use objects of non-uniform size, and little work has been done on packing collections of such objects.

We should note that it is difficult to draw direct comparisons between our results and those of Wilson et al. [7], as their model enforces strict spatial constraints on the movement of ants and objects. In addition, Hartmann [8] presents results only in the context of genetic algorithm fitness evaluations, with no individual breakdowns for each metric, so direct comparisons are again difficult (although his paper does use the same separation and shape algorithms as the those used by Wilson *et al.* [7] and ourselves). Nonetheless, the metrics provide a useful standardised framework for performance analysis.

Three sets of experiments were carried out to investigate the algorithm's performance, given the following initial configurations:

1) Equal numbers of each object, randomly distributed (as in [7], [8]),
2) Unequal numbers of each object, randomly distributed (as observed in nature [6]),
3) Pre-sorted, equally-sized clusters of objects.

A. Equal Object Numbers

The first set of experiments replicated the initial conditions described in [7]: 15 objects of each type, randomly distributed across the surface, with 6 ants. The average separation and shape scores for 50 initial configurations were 11.85% and 48.21% respectively. The results obtained are depicted in Table I, with the best figures obtained highlighted in bold. The radial displacement distributions and a typical final pattern are depicted in Figure 4.

The figures of 79.52% and 70.88% for separation and shape respectively compare well with the figures of 59% and 68.5% obtained by the leaky integrator of Wilson *et al.* [7] (noting that their simulation includes extra spatial constraints and uses objects of uniform size, whilst ours has no such constraints but handles objects of different sizes).

TABLE I
RESULTS FOR 15 OBJECTS OF EACH TYPE WITH 6 ANTS, AVERAGED OVER 50 RUNS FOR EACH ENERGY VALUE.

Energy	*Separation*	*Shape*
250	51.38	56.49
500	61.71	60.17
750	71.09	64.65
1000	77.19	68.21
1250	77.14	68.96
1500	**79.52**	**70.88**
1750	79.0	70.86

B. Unequal Object Numbers

The aim of the second set of experiments was the assess the algorithm against the type of configuration observed in actual *Temnothorax* nests; that is, where there are many more small brood items than large (i.e., older) items [6] (see Figure 1). In these experiments, we randomly distributed 40 small objects, 20 medium objects and 10 large objects. In order to retain the ant-to-object ratio used in the previous set of experiments, we used 10 ants in this set. The average separation and shape scores for 50 initial configurations were 17.16% and 48.25% respectively. The results obtained are depicted in Table II, with the best figures obtained highlighted in bold. The radial displacement distributions and a typical final pattern are depicted in Figure 5.

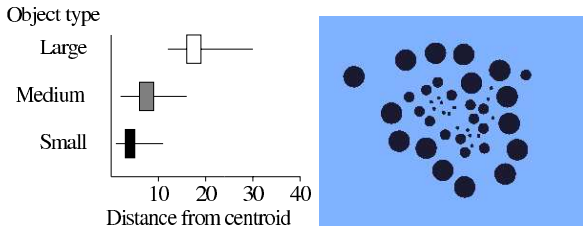

Fig. 4. Equal numbers of objects. (Left) Radial displacement. Minimum and maximum distances are represented by the line, and the interquartile range by the box. (Right) Typical pattern obtained.

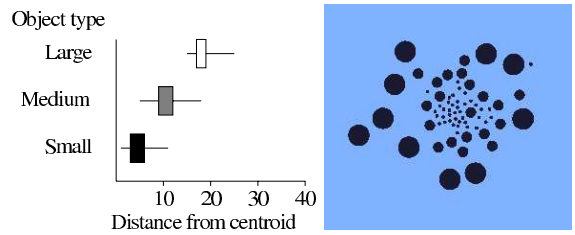

Fig. 5. Unequal numbers of objects. (Left) Radial displacement. (Right) Typical pattern obtained.

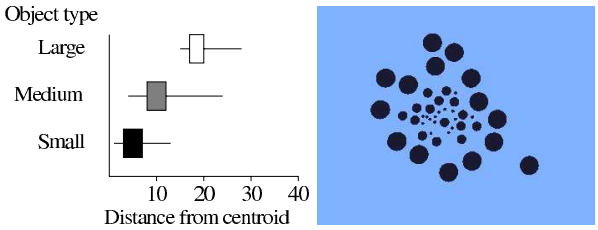

Fig. 6. Pre-sorted objects. (Left) Radial displacement. (Right) Typical pattern obtained.

The algorithm clearly performs best when applied to distributions of objects that roughly match those observed in nature. The high separation score of 93.04% is in general partly due to the observed creation of a large, densely-packed core of small objects at the centre of the structure. Once built, this core is rarely disturbed by the ants, and sorting only occurs in the outer bands.

TABLE II

RESULTS FOR |SMALL|=40, |MEDIUM|=20, |LARGE|=10 WITH 10 ANTS, AVERAGED OVER 50 RUNS FOR EACH ENERGY VALUE.

Energy	*Separation*	*Shape*
250	76.86	75.10
500	87.00	81.30
750	**93.47**	**84.62**
1000	93.04	84.37

C. Pre-sorted Clusters

The aim of the second set of experiments was the assess the algorithm's ability to perform annular sorting of objects that were pre-sorted into piles. We created three piles, each one consisting of 15 objects of a particular size randomly clustered around a fixed point (Figure 7). As in the first experiment, 6 ants were used. The separation and shape scores for initial configurations are clearly meaningless in this context, so we omit them here. The results obtained are depicted in Table III, with the best figures obtained highlighted in bold. The radial displacement distributions and a typical final pattern are depicted in Figure 6.

Our studies show that the algorithm is able to convert a pre-sorted configuration into one that is sorted in an annular fashion. Given sufficient energy, there is little difference in performance in sorting either pre-sorted or randomly distributed configurations. Observation of the algorithm shows that, in general, the ants form two clusters of roughly equal size and composition (Figure 7). These are gradually merged into a single structure which is then refined in terms of shape and separation. It is important to note that no modifications (either to the model code or to the parameters) were necessary in order for these results to obtained. This suggests that the model is robust and capable of dealing with a variety of initial configurations.

V. CONCLUSIONS

This study has demonstrated a minimalistic model that can consistently sort objects into annular clusters. It is the first known annular sorting algorithm that can deal both with objects of different sizes and pre-sorted configurations. There

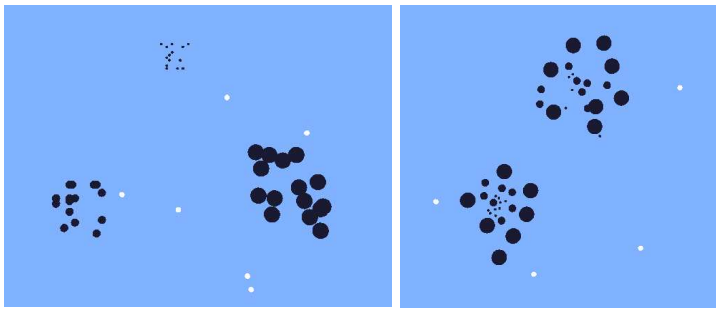

Fig. 7. (Left) Pre-sorted objects. (Right) Intermediate stage in global sorting of locally pre-sorted objects.

TABLE III
RESULTS FOR 15 PRE-SORTED OBJECTS OF EACH TYPE WITH 6 ANTS, AVERAGED OVER 50 RUNS FOR EACH ENERGY VALUE

Energy	*Separation*	*Shape*
250	41.71	55.46
500	64.14	62.82
750	75.19	67.02
1000	**77.23**	**68.95**
1250	76.67	67.87

are real world examples where this might be useful, particularly when one has objects that may change over time and require different amounts of separation. Our algorithm uses only stochastic ant behaviour and a brood item "attraction-repulsion" mechanism, and is the simplest model of annular sorting to date. However, some of the aspects that set this study apart from others such as Wilson *et al.* [7] also make it harder to create a physical implementation (perhaps using mobile robots). In particular the model relies on the fact that agents are able to pull poorly placed objects out of the centre of a cluster without disturbing other objects or risking collision, something that is hard to do with simple robotics. These difficulties will become less significant as the state of the art advances. Another significant issue is that of algorithm termination. We currently use an energy term to "wind down" the simulation, but it has been suggested [15] that a more satisfactory solution may be an environmentally-driven cue, based on *stigmergic* responses [16] to changes already made. Future work will study this in further detail, as well as the broader biological significance of the "attraction-repulsion" mechanism.

ACKNOWLEDGEMENTS

The authors are grateful to Jon Klein, Chris Melhuish and Ana Sendova-Franks for useful advice and suggestions.

REFERENCES

[1] S. Camazine, J.-L. Deneuborg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-organization in biological systems*. Princeton University Press, 2001.
[2] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.
[3] M. Dorigo and T. Stützle, *Ant colony optimization*. MIT Press, 2004.
[4] J.-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien, "The dynamics of collective sorting: Robot-like ants and ant-like robots," in *Proceedings of the First International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 1*, J.-A. Meyer and S. Wilson, Eds. MIT Press, 1991, pp. 356–365.
[5] J. Handl, J. Knowles, and M. Dorigo, "Ant-based clustering: a comparative study of its relative performance with respect to *k*-means, average link and 1d-som," technical Report TR/IRIDIA/2003-24, IRIDIA, Universite Libre de Bruxelles, July 2003. http://www.handl.julia.de.
[6] N. Franks and A. Sendova-Franks, "Brood sorting by ants: distributing the workload over the work-surface," *Behavioural Ecology and Sociobiology*, vol. 30, pp. 109–123, 1992.
[7] M. Wilson, C. Melhuish, A. B. Sendova-Franks, and S. Scholes, "Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies," *Autonomous Robots*, vol. 17, pp. 115–136, 2004.
[8] V. Hartmann, "Evolving agent swarms for clustering and sorting," in *Proceedings of the 2005 conference on Genetic and Evolutionary C omputation (GECCO 05)*. ACM Press, 2005, pp. 217–224.
[9] S. Scholes, M. Wilson, A. B. Sendova-Franks, and C. Melhuish, "Comparisons in evolution and engineering: the collective intelligence of sorting," *Adaptive Behavior*, vol. 12, no. 3–4, pp. 147–159, 2004.
[10] A. Vik, "Evolving annular sorting in ant-like agents," in *Proceedings of the European Conference on Artificial Life (LNAI 3630)*. Springer-Verlag, 2005, pp. 594–603.
[11] A. Scheidler, D. Merkle, and M. Middendorf, "Emergent sorting patterns and individual differences of randomly moving ant like agents," in *Proceedings of the 7th German Workshop on Artificial Life (GWAL-7), Jena, Germany*, 2006.
[12] S. Gueron, S. A. Levin, and D. I. Rubenstein, "The dynamics of herds: from individuals to aggregations," *Journal of Theoretical Biology*, vol. 182, pp. 85–98, 1996.
[13] J. H. Tien, S. A. Levin, and D. I. Rubenstein, "Dynamics of fish shoals: identifying key decision rules," *Evolutionary Ecology Research*, vol. 6, pp. 555–565, 2004.
[14] J.Klein, "breve: a 3D simulation environment for the simulation of decentralized systems and artificial life," in *Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems*. The MIT Press, 2002, pp. 329–334.
[15] C. Melhuish, personal communication, July 2005.
[16] O. Holland and C. Melhuish, "Stigmery, self-organisation, and sorting in collective robotics," *Artificial Life*, vol. 5, pp. 173–202, 1999.