Swarm Robotics

Clustering and Sorting

By Andrew Vardy
Associate Professor
Computer Science / Engineering
Memorial University of Newfoundland
St. John’s, Canada
Deneubourg et al (1990)

- Inspired by observations of ant behaviours that create global order through local action
- Dead ants moved into "cemetery clusters" that aggregate over time
- Nest contents organized into distinct piles
- Deneubourg et al’s model:
  - Agents walk randomly and pick-up or deposit objects as a probabilistic function of local object density
Agents measure density by maintaining a short-term memory and counting the number of recent object appearances.

\[ P_{pu} = \left( \frac{k_1}{k_1 + \text{density}} \right)^2 \]

\[ P_{de} = \left( \frac{\text{density}}{k_2 + \text{density}} \right)^2 \]
Fig. 1. Clustering after 1, 100000 and 2000000 steps. 100 ALRs, 1500 objects, $k^-=0.1, k^+=0.3, m=50, e=0$, space=290x200 points. Small evenly spaced clusters rapidly form, and later merge into fewer larger clusters.

Fig. 2. Clustering in a colony of *Pheidole pallidula*. 4000 corpses were placed on a 50x50cm arena, and photos taken at time 0, 20 and 68 hrs. Small evenly spaced clusters rapidly form, and later merge into fewer larger clusters.
Clustering and Sorting

- Clustering: One object type
- Sorting: More than one object type
- Objects can be organized in different ways: e.g. concentric rings or patches
- We focus on patch sorting: “Grouping two or more classes of objects so that each is both clustered and segregated, and each lies outside the boundary of the other” (Melhuish et al, 1998)
This shows the extension of Deneubourg's model to handle multiple object types (i.e. sorting).

FIGURE 4.4  Simulation of the sorting model. (a) Initial spatial distribution of 400 items of two types, denoted by o and +, on a 100 x 100 grid. (b) Spatial distribution of items at t = 500,000. T = 50, k_1 = 0.1, k_2 = 0.3, 10 agents. (c) Same as (b) at t = 5,000,000.
Beckers, Holland, and Deneubourg (BHD) wrote a paper detailing their experiments in swarm robotic clustering.

Robots in the BHD experiment act according to the Deneubourg et al (1990) model, but the pick-up / deposit behaviour is implicit.

Figure 1: Robot equipped with a gripper for object Gathering. Experiments were carried out with from 1 to 5 robots of the same type,
C-shaped gripper passively collects pucks

Sensors and behaviours:

- Infrared sensors to detect obstacles (walls, other robots)
  - Behaviour: Triggers random turn away from obstacle
- Microswitch attached to gripper detects that gripper is pushing against three or more pucks
  - Behaviour: Triggers backup, then a random turn, resulting in the pucks being left behind (i.e. deposited)
- If no behaviour is triggered, the robot just moves straight

FIGURE 4.15  Robot 1, because it has only two pucks in its C-shaped gripper, continues to move in a straight line. Robot 2, with three pucks in its gripper, is encountering a cluster: its microswitch is activated by the gripper, so that the robot leaves the pucks next to the cluster.
Figure 2. The initial setup (a) and time evolution of a typical experiment involving a group of three robots. Phase 1 (b), occurring after approximately 10 min, is characterised by a large number of small clusters containing from 1 to 10 pucks. In Phase 2 (c), some clusters grow rapidly and the environment becomes more heterogeneous. Finally, Phase 3 (d) is characterised by the competition between a small number (2 – 3) of large clusters and evolves towards the clustering of all objects in one pile.
Motivation: Why Study Clustering and Sorting

- Modelling of sorting and clustering behaviours in nature
- Possible applications:
  - Cleaning (aggregating waste material)
  - Recycling
  - Preparing resources for assembly
The Assumption of Extreme Simplicity

Several other researchers have pursued further developments on the idealized model of Deneubourg et al. (1990) or BHD (1994) under the following assumptions:

- Agents are extremely simplistic with no long-term memory, no capacity for navigation, and a lack of reaction to distal stimuli.
- It has become evident that insects such as ants and bees do not share these limitations.
Using Vision

* Deneubourg’s et al’s model and related variants (e.g. BHD) exhibit random motion and a perceptual radius of zero

* The events of interest (pick-ups and deposits) happen by chance

* The robot’s view of nearby pucks can be processed to yield a list of clusters of each type

* We can also determine the type of the carried puck
Modified SRV-1 robots (12.5 x 10.8 cm) with forward-facing fisheye cameras and passive grippers, suitable for carrying (and viewing) one puck.
Fig. 2 This figure shows the view from one of our SRV-1 robots (a) and a simulated robot (b). In (a) the robot’s raw view, colour segmented image, and local map are shown from left-to-right. In (b) an overhead view of the simulator is shown, the simulated robot’s raw view (no colour segmentation is required), and the local map.

Some additional processing is applied to the information stored in the local map. Each robot can see within its own gripper and may also be able to see within the grippers of other nearby robots. To prevent one robot from attempting to collect a puck already held by another, we remove from the list of pucks any that lie within a threshold distance in the image plane of another robot. The effect of this rule is visible in Figure 2(a) in that the red cells within the gripper of the other robot are not extracted as a puck (they are not circled). It is also apparent from this figure that the other robot is partially classified as a robot (blue), but also partially classified as an obstacle (black). This is simply because parts of the robot are difficult to paint, such as the black rubber treads. Also, we take special note of puck colours within the robot’s own gripper. If the fraction of puck colours within the gripper is high enough then we assert that a puck is being carried. A puck at the position of the gripper is added to the local map in this case. The carried puck, just like other pucks, can belong to a cluster. If so, we know the robot has made contact with this cluster. This is a significant event for the sorting methods discussed below.

Clusters extracted from the local map are denoted as $G(i,j)$ where $i$ is the index of the cluster and $j$ is the object type. These clusters are graphs $G(i,j) = (V(i,j), E(i,j))$ where $V(i,j)$ corresponds to the node set (i.e. the perceived pucks) and $E(i,j)$ is the set of edges between nodes. The algorithms described below act based on the number of pucks in a cluster. Therefore we define the size of a cluster as the number of nodes: $\text{size}(G(i,j)) = |V(i,j)|$. The smallest and largest clusters of type $j$ Input Image

- **Pucks:**
  - Connected blobs of similar coloured pixels

- **Clusters:**
  - Sets of nearby pucks with inter-puck distances below a threshold. A puck is in a cluster if it is close enough to any other puck in the cluster.
  - Homogeneous by definition

Colour Segmented

Local Map:
New Algorithm: ProbSeek

- We can now react to clusters based on their size using the following heuristics:
  - If carrying puck
    - Consider depositing at the largest matching cluster in view
  - If not carrying puck
    - Consider collecting a puck from the smallest cluster in view
- "Consider" means apply a probabilistic decision rule based on the candidate cluster's size
The robot is not carrying a puck

It would consider selecting the solitary red puck as a pick-up target

If carrying a red puck, it would consider the cluster of two red pucks as a deposit target

Possible results of the pick-up/deposit attempt handled through the state machine...
Cache Consensus: Rapid Object Sorting by a Robotic Swarm

Target acquired
Puck carried (unexpected)
Puck carried
Lost target
Puck lost (unexpected)
Target acquired
Cluster contacted
Puck lost (unexpected)
Lost target
Timeout
Timeout
Timeout
Start

Fig. 5

Finite state machines for ProbSeek and CacheCons. The HOMING and EXILE states are highlighted by enlarged font in (b).

When the carried puck becomes part of a cluster, due to the way in which clusters are extracted, two pucks can only belong to the same cluster if they are of the same type. If this occurs, the state transitions to DE_PUSH.

States DE_PUSH, DE_BACKUP, and DE_TURN are functionally identical to the states PUSH, BACKUP, and TURN defined for BHD.

ProbSeek shares several features in common with the method presented in our previous work (Vardy 2012), in particular targeting the smallest cluster in view for possible pick-up and the largest cluster for possible deposit. This feature was found to accelerate clustering performance over the method of Beckers et al.

The main difference from the previous method is the incorporation of Deneubourg et al.'s probabilistic heuristics. For an unladen robot, the previous method would always target the smallest cluster in view regardless of the size of that cluster. Incorporating the Deneubourg et al. heuristic means that small clusters will be targeted for pick-up more readily than larger clusters. This is consistent with the goal of convergence to one cluster. Similarly, in the case of deposit, ProbSeek is more likely to target a larger cluster than a smaller one. The other significant difference is the method of target selection.

FSM for ProbSeek Algorithm

Technical Details
Supplementary Video:

'Rapid object sorting by a robotic swarm via cache consensus’

Trial 1 / 3
Localization

* ProbSeek provides a significant improvement in sorting performance but maintains no memory of past clusters

* The ability to return to significant places in the environment can be achieved in many ways:
  * Visual homing
  * Map-based localization
  * Cheating (e.g. GPS, overhead camera)
New Algorithm: CacheCons

- CacheCons is based on ProbSeek with the following main modifications:
  - Cache Points are maintained to represent the largest clusters seen
  - When a puck is collected, the robot homes to the cache point and deposits
  - There remains no communication between robots; cache consensus is emergent
ProbSeek

CacheCons

Technical Details

Fig. 5

Finite state machines for ProbSeek and CacheCons. The HOMING and EXILE states are highlighted by enlarged font in (b). When the carried puck becomes part of a cluster. Due to the way in which clusters are extracted, two pucks can only belong to the same cluster if they are of the same type. If this occurs the state transitions to DE_PUSH.

States DE_PUSH, DE_BACKUP, and DE_TURN are functionally identical to the states PUSH, BACKUP, and TURN defined for BHD.

ProbSeek shares several features in common with the method presented in our previous work (Vardy 2012), in particular targeting the smallest cluster in view for possible pick-up and the largest cluster for possible deposit. This feature was found to accelerate clustering performance over the method of Beckers et al. The main difference from the previous method is the incorporation of Deneubourg et al's probabilistic heuristics. For an unladen robot the previous method would always target the smallest cluster in view regardless of the size of that cluster. Incorporating the Deneubourg et al heuristic means that small clusters will be targeted for pick-up more readily than larger clusters. This is consistent with the goal of convergence to one cluster. Similarly in the case of deposit, ProbSeek is more likely to target a larger cluster than a smaller one. The other significant difference is the incorporation of Deneubourg et al's probabilistic heuristics.
Details

* ProbSeek and CacheCons:
  * Avoidance of robots, walls, and non-targeted clusters: VFH+

* CacheCons:
  * Memory of cache sizes: \[ m_j' = \max(m_j, \text{size}(C_j)) \]
  * Caches for different puck types separated by at least 50 cm (if conflict, the larger cache is kept)
Experimental Setup

187 x 187 cm

Visual markers on robots tracked from above (cheating!)

187 x 187 cm

Painted Pucks

Rounded Corners

Experimental Setup
Cache Consensus: Rapid Object Sorting by a Robotic Swarm

Fig. 2

This figure shows the view from one of our SRV-1 robots (a) and a simulated robot (b). In (a) the robot’s raw view, colour segmented image, and local map are shown from left-to-right. In (b) an overhead view of the simulator is shown, the simulated robot’s raw view (no colour segmentation is required), and the local map.

Some additional processing is applied to the information stored in the local map. Each robot can see within its own gripper and may also be able to see within the grippers of other nearby robots. To prevent one robot from attempting to collect a puck already held by another, we remove from the list of pucks any that lie within a threshold distance in the image plane of another robot. The effect of this rule is visible in Figure 2(a) in that the red cells within the gripper of the other robot are not extracted as a puck (they are not circled). It is also apparent from this figure that the other robot is partially classified as a robot (blue), but also partially classified as an obstacle (black). This is simply because parts of the robot are difficult to paint, such as the black rubber treads. Also, we take special note of puck colours within the robot’s own gripper. If the fraction of puck colours within the gripper is high enough then we assert that a puck is being carried. A puck at the position of the gripper is added to the local map in this case. The carried puck, just like other pucks, can belong to a cluster. If so, we know the robot has made contact with this cluster. This is a significant event for the sorting methods discussed below.

Clusters extracted from the local map are denoted as $G(i)_{j}$ where $i$ is the index of the cluster and $j$ is the object type. These clusters are graphs $G(i)_{j} = (V(i)_{j}, E(i)_{j})$ where $V(i)_{j}$ corresponds to the node set (i.e. the perceived pucks) and $E(i)_{j}$ is the set of edges between nodes. The algorithms described below act based on the number of pucks in a cluster. Therefore we define the size of a cluster as the number of nodes: $\text{size}(G(i)_{j}) = |V(i)_{j}|$. The smallest and largest clusters of type $j$
3.4 Performance Metrics

Anumber of differentmetrics have been proposed to assessdistributed clustering and sorting algorithms. These include number of clusters (Beckers et al, 1994; Maris and Boeckhorst, 1996), size of the largest cluster (Beckers et al, 1994), mean cluster size (Maris and Boeckhorst, 1996; Martinoli et al, 1999), and spatial entropy (Bonabeau et al, 1999). Melhuish et al (1998) introduced a metric for sorting algorithms whereby completion is declared when some fixed percentage of all pucks lie within the largest cluster of their particular colour. We utilize percentage completion as our base metric.

The number of objects (pucks) of type \( j \) is \( n_j \). Thus, the total number of pucks is \( n = \sum j n_j \). Equation 2 defined \( \Lambda_j \) as the largest cluster of type \( j \) in the robot's local map. For analytical purposes we assume access to the global map which describes all clusters with respect to world coordinates. Clusters are defined in the global map in the exact same manner as for a robot's local map. The largest cluster of type \( j \) in the global map is denoted as \( \Lambda^*_j \) and percentage completion (PC) is defined as follows.

\[
PC = 100\% \cdot \frac{\sum_j \text{size}(\Lambda_j)}{n}
\]

One important metric derived from percentage completion is the number of steps required to reach a particular level of completion. We track the first time the level of percentage completion reaches a target threshold. Since 100% completion is not reached by all tested methods, we use a less ambitious threshold of 50% completion. We refer to this measure of steps-to-completion as STC. It is possible for a sorting method to reach the target completion threshold, but then to degrade in performance. To capture this phenomenon we use a time-weighted sum over

**Performance Metric**

**Technical Details**

- Percentage Completion
- Largest cluster of type \( j \)
Fig. 7 Plots of percentage completion versus time step while varying the number of puck types. The mean value for each data set is indicated by a heavy trace, surrounded by a shaded region. The extent of the shaded region is ±1.96 standard errors. Thus, these shaded regions correspond to 95% confidence intervals for the mean.

4.2.1 One Object Type

When there is only one object type, the sorting problem becomes identical to the clustering problem. In this case the methods tested included BHD, ProbSeek, and CacheCons. The upper left plot in Figure 7 suggests a substantial difference in performance between these three methods. Snapshots from the simulation shown in Figure 8 further support the observation that CacheCons very quickly reaches and maintains convergence, while ProbSeek and BHD are unable to reach the same level of completion within the allotted time. This observation is further confirmed with statistical tests. First, we considered the two metrics, TWC and STC. Figure 9 shows the distribution of TWC and STC values. The D’Agostino & Pearson normality test indicates that the TWC data for ProbSeek and CacheCons do not follow a normal distribution, however the STC data for all three methods does appear to follow a normal distribution. Focussing our analysis on STC allows us to use the standard parametric tests which have greater statistical power than non-parametric tests. We performed a repeated measures ANOVA which indicated that the mean STC values are all drawn from different distributions (p<0.0001). Applying Tukey’s multiple comparisons test revealed that all differences were significant. That is, the mean STC for BHD is significantly greater than for ProbSeek and CacheCons and the mean STC for ProbSeek is significantly greater than for CacheCons (in all cases p<0.0001).
Supplementary Video: "Rapid object sorting by a robotic swarm via cache consensus”

Trial 1 / 3
Conclusions

- Dropping the extreme simplicity assumption doesn’t curtail potential correspondence with biology
- The ability to home to remembered locations enables vastly improved sorting performance
References


