Division of Labour in Simulated Ant Colonies Under Spatial Constraints

Jesper Bach Larsen

EvAlife Group, Dept. of Computer Science, University of Aarhus, Denmark
jesper@larsen.cc, www.larsen.cc/jesper

Abstract. Division of labour in ant societies is a kind of role model for distributed problem solving. They are so because ants are simple, non-cognitive, distributed and autonomous and yet they solve an optimisation problem that is very complex and dynamic. This is very desirable in computer science, but as of yet not much research has gone into explaining the underlying mechanisms of division of labour. The venue in this paper is to, by means of evolutionary algorithms, find the implications spatial constraints play in the division of labour in a foraging task of virtual ants. The ants differ only in size, and size implies constraints regarding to ease of movement in an environment where obstacles of other ants and clay exists. The results show that spatial constraints do play a major role in the job-task division evolved, in that test setup with increasing constraints exhibited division of labour in that ants of different sizes occupy different spatial areas of the task domain. In the process, we have evolved the behaviour of the ants that underlie the division of labour. This was done via mapping functions and motivation networks.

1 Introduction

The effectiveness of an ant society can be highly contributed to the fact that division of labour occurs in the colony. Seen from the outside the foraging task seems to be centrally coordinated, or at least controlled by some stringent and predetermined rules or patterns [3]. Yet the colony is not directed by some hive mind, and the underlying mechanisms of division of labour is of yet mostly unexplained. As inspiration for division of labour in computer systems, ant societies are brilliant. They are so because ants are simple entities, non-cognitive, distributed and autonomous and yet the overall colony performs fault tolerant, flexible and seemingly intelligent. Biology research explains division of labour as evolutionary optimisation, and their models are top-down aggregated approaches based on observations. This is the case in age-polyethism in which the age is the major determinant of subtask assignment [3,4]. Many studies have shown strong evidence against this postulated correlation, by showing great flexibility in task assignment under radical changes in the population, whereby older ants would revert to tasks previously performed, and younger ants would accelerate the transition to new tasks as well [1,2]. Other mechanisms must thus underlie the mechanisms of division of labour. In this paper we shall pursue the theory of physical polymorphism in which the morphology is used as the determining parameter of task assignment. The importance of morphology towards division of labour is investigated. The importance seems plausible since fit to tasks depend on morphology. In our experiments we have tailored a virtual ant environment with 2 physical different worker castes and a foraging task that requires 2 opposing physical characteristics of the ants. A larger body size implies certain problems when moving in narrow places, as is the case in the constructed nest-structure. The big ants have an advantage where the spatial constraints are low, as in the food collecting areas, whereas the smaller ants move slower, and have a disadvantage here. In the nest the smaller ants have the advantage of being able to move more freely around without colliding into nest-walls, whereas the big ants just merely can move around. We thus seek a spatial pattern in the foraging task of the big versus the small ants. As such it is the emergent effects in the foraging task under spatial constraints that we seek. An Evolutionary Algorithm evolves the behaviour used to achieve this. From a computer science perspective, our aim is to obtain a better understanding of how autonomous distributed agents can cooperate in a foraging task in an environment where only local knowledge is available, and where a division of the task is needed, and where no centralized mechanisms can be used to coordinate this division.

This paper is organized in the following way. Chapter 2 describes the AntDOL model, which underlies our simulations. In chapter 3 the experiments performed and results obtained are presented. A discussion of the implications is found in chapter 4, as is also shortcomings and ideas to pursue in further research.

2 The AntDOL Model

The following model is based on a simulation-framework developed in cooperation with Ilja Booij, Leiden University, Holland.

The model for division of labour in virtual ants, hereafter called AntDOL, is an evolutionary algorithm where the population of individuals is a population of ant colonies. As usual fitness is calculated for each of these individuals, and the fittest individuals are allowed to recombine into new individuals for the following generation. To calculate the fitness of an individual (an ant colony) the colony is simulated for a fixed number of time steps, and the number of food items foraged during the simulation is used for the fitness calculation. Each individual has a virtual gene that encodes the behaviour of all ants in the colony. Statistical data is collected from the simulations, which are stored for later analysis. The task is to find the optimal colony in regard to the fitness-function, and then retrospective to observe statistical and spatial emergent patterns of division of labour, i.e. do the different castes have a specific spatial placement.

2.1 The Ant Colony

The colony is represented as a discrete grid of locations that can hold specific objects. There can be 3 different types of objects on a location: ant, food and obstacle. There can be only one ant at a location, and only if no obstacle is placed there. There can be any number of food objects on a single location. The grid uses a 9-cell Moore Neighbourhood, so that each ant, from any interior location, can move in 8 directions.

Movement is restricted to be within borders, i.e. no wrapping, and ants can only pass locations without obstacles and ants.

2.1.1 Ants

Ants come in 2 sizes. The small cover 4x4 locations, and the large cover 6x6 locations. The speed with which the ant can move is defined as $\lfloor AntSize/2 \rfloor$, i.e. a 4x4 ant can move 2 locations at a time, and a 6x6 can move 3 locations at a time. Ants have sensors and actions that enable them to respond to the local environment. Ants can only sense and influence the closest vicinity of the environment. All sensors are real valued in the interval [0,1]. The ants can retrieve data from the environment by these sensors only. The following sensors are implemented in the system. Sensor senseFood returns a direction to a possible food-source in the immediate vicinity of the ant. Food in the direction of movement is preferred. Sensor senseAnt returns a weighted score of the presence of other ants in the direction of movement. Closer ants are weighted heavier. Sensor senseClay returns a weighted score of the presence of clay in the direction of movement. Closer clay objects are weighted heavier. Sensor sensePointingHome returns a direction to the nest (1/360 degrees). Sensor senseCarryFood returns 1 if the ant is carrying food, and returns 0 otherwise. Sensor senseFrustrated returns a frustration level, which is incremented when intended actions are blocked (i.e. by obstacles or other ants), and decreased with time. The ant can only make changes to the environment and to itself by actions. The following actions were implemented in the system. Action MoveForward moves the ant $\lfloor AntSize/2 \rfloor$ steps in the current direction. If the path is blocked by obstacles, ants or borders, the ant is moved only the possible number of locations. Action PickupFood makes the ant pickup a random food object within the nearest vicinity. Food is not consumed, just carried. There is no difference in carrying capacity of the ants. Action DropFood makes the ant drop a carried food item at the current location. Action TurnLeft makes the ant turn anti-clockwise to the next Moore Neighbourhood location. Action TurnRight makes the ant turn clockwise to the next Moore Neighbourhood location. The sensors and actions are coupled together via behaviours, which are described next.

2.1.2 Behaviours and Decision Making

The virtual ants are controlled by a motivation-network, which is a decision-making model of animals that is based on response-curves. As such it is a model that builds upon the idea of response-thresholds by Bonabeau, merely with several thresholds for each behaviour, and the response-curves used are different [0]. In a motivation network a number of behaviours compete for execution based on a maximum score. For sake of planning in the foraging task, we have constructed a number of high level behaviours, that are solely composed of the sensors, and the basic actions presented earlier. The following behaviours were implemented in the system: In behaviour findFood, if food can be sensed, the ant will go to that location. If no food can be sensed, the ant will wander almost randomly around, with a preference for moving away from the nest. If food is found, it is not picked up automatically. In behaviour findHome, if the ant is within the nest and it carries food, it will drop this food item. If the ant is away from the nest it will follow the nest sensor towards the nest, and avoid obstacles underway by use of the ant and clay sensors. In behaviour takeFood, the action PickupFood is called. In behaviour dropFood, the action DropFood is called.

The behaviours each used the following sensors: In behaviour findFood and findHome, the senseCarryFood sensor was used. In behaviour takeFood, the senseFood, senseCarryFood, senseAnt and senseFrustration was used. In behaviour dropFood, the sensors senseFood, senseCarryFood, senseAnt, sensefrustration and senseClay was used. The motivation-network is implemented such that each sensor S_i of a behaviour B_k is mapped to a real value, $f_{k,j}(S_i)$, in the interval [0,1] by a mapping function $f_{k,j}$. The mapped sensor value, or the average of all the mapped sensor values (of the behaviour uses more than one sensor) is called the score of the behaviour. The behaviour with the highest average score is selected for execution at each time step. Our mapping functions are 5 degree polynomials discretized by 6 points in the interval [0,1]. Evaluation of a point in the interval [0,1] is done by interpolation. This coupling of sensors to behaviours via mapping-functions enables the ant to respond in a very differentiated way based on the shape of the mapping functions. The mapping functions (the 6 real numbers) are encoded in virtual genes as real numbers. Please notice that even though all ants in a give colony share the same genes, their behaviour is different. This is due to differences in sizes (which relates to obstacle avoidance), and local sensor stimuli, that e.g. eventually will influence the frustration sensor differently. The reason that we have hand-coded high level behaviours and not evolved them is due to the emphasis on the spatial patterns of division of labour, and only secondary the behaviour itself. See figure 1 for a model of the motivation network.

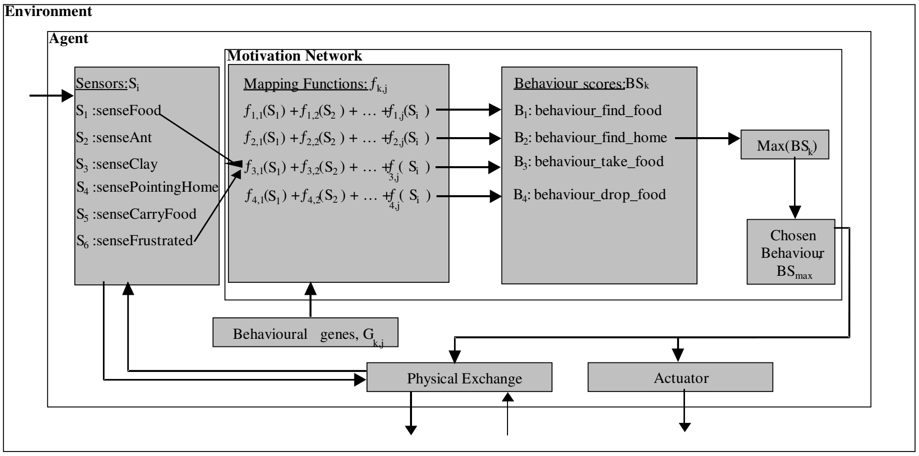

Environment
Agent
Motivation Network
Sensors: S_i
S_1 :senseFood
S_2 :senseAnt
S_3 :senseClay
S_4 :sensePointingHome
S_5 :senseCarryFood
S_6 :senseFrustrated
Mapping Functions: $f_{k,j}$
$f_{1,1}(S_1) + f_{1,2}(S_2) + \ldots + f_{1,j}(S_i)$
$f_{2,1}(S_1) + f_{2,2}(S_2) + \ldots + f_{2,j}(S_i)$
$f_{3,1}(S_1) + f_{3,2}(S_2) + \ldots + f_{3,j}(S_i)$
$f_{4,1}(S_1) + f_{4,2}(S_2) + \ldots + f_{4,j}(S_i)$
Behaviour scores: BS_k
B_1: behaviour_find_food
B_2: behaviour_find_home
B_3: behaviour_take_food
B_4: behaviour_drop_food
Max(BS_k)
Chosen Behaviour BS_{max}
Behavioural genes, $G_{k,j}$
Physical Exchange
Actuator

Fig. 1. Motivation Network. The coupling between sensors and behaviour through mapping functions which are encoded as artificial genes.

2.2 The AntColony EA

As noted, the 6 sample-points of every mapping function, are encoded in a gene-string that is used by an individual (a colony) in the EA. Because of this all ants in the colony share the same gene. This model uses 11 mapping functions, and thus maximizes the fitness over 66 real-variables. Mutation and crossover operators are applied to a hand coded gene-seed to initialise the population. The fitness of a colony is calculated on the number of food items the ants drop in the center of the nest (zone 0) relative to the number of ants and the number of simulation steps. Because the simulation is highly stochastic, a sliding average of 5 generations was used for fitness calculation (see Equation 1 below). Please note that fitness is calculated on the aggregate result of a whole colony simulation, not on the individual ant. The individual ant does not exist in the EA model used.

$$\text{Fitness(Colony)} = c * (\text{Foods delivered last 5 generations}) / (\text{Number of Ants} * \text{Simulation steps in colony} * 5) \quad \textbf{(1)}$$

To further prevent stagnation due to stochastic influence we also copy the best colony to the next generation, unaltered (elitism). An outline for the EA algorithm can be seen below. As such the EA works as usual, but with the little twist that the fitness evaluation of an individual is a simulation of the colony. The colony is simulated according to the genes, which encode the behavior. An outline for the EA algorithm can be seen below.

```
AntColonyEA()
{
  t=0
  Initialise P(t)

  for a fixed number of generations do
  {
    t=t+1
    for each Colony in P(t-1) do
    {
      Simulate Colony according to Genes
      Calculate fitness of Colony
    }

    TournamentSelect (n-1) colonies from P(t-1) to P(t)
    Mutate all colonies in P(t)
    Recombine all colonies in P(t)
    Copy Elitist Colony from P(t-1) to P(t)
  }
}
```

Fig. 2. The AntColony evolutionary algorithm. The TournamentSelect uses size 2. The variable ‘t’ is used as generation counter, and ‘P(t)’ designates the Population set at time ‘t’.

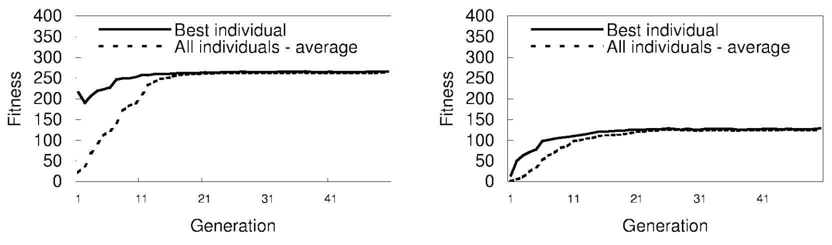
3 Experiments and Results

Different experiments were performed with varying simulation-parameters. For the AntColonyEA a population of 50 colonies was chosen. This might not be a very big population compared to the complexity of the solution-space, but the restriction had to be made due to the very cpu intensive nature of this setup. All AntColonyEA simulations were run for 50 generations. The mutation rate used was $p_m = 0.7$, with the time dependent variance σ^2(generation) = 1/(1+generation)., which is also called simulated annealing For the crossover operator we used $p_c = 0.7$, and it was implemented as a swap operator, i.e. genes were swapped between individuals and not arithmetically combined. On the colony/individual level we used 60 ants in all, 30 of size 4x4, and 30 of size 6x6. Simulations with different ratios between sizes were also done; these are reported briefly in the results section. Each colony was simulated for 2000 time steps. The grid world is of size 300x300 locations. Food was placed randomly in circles in the perimeter of the world, and replenished all the time, so lack of food is not an issue in the optimization. One set of experiments was performed without spatial constraints, and another one with spatial constraints. These sets are then subsequently compared. The spatial constraints in the experiments were small clay objects spaced evenly around the center of the nest with the perimeter of the nest being a little more open than the center. The nest structure is open enough for the big ants to move around, albeit with a relative high probability of colliding with clay objects. The center of the nest, called zone 0 is the destination for the foraging task, and food items dropped there are recorded as delivered. The rest of the nest is called zone 1 and extends 1/3 out of the distance to the world-border. The rest of the world is zone 2. For later comparison around 0,7% of the total area is in zone 0, 9,7% is in zone 1, and the rest 89,6% in zone 2. Please see figure 5 for a picture of the simulation environment.

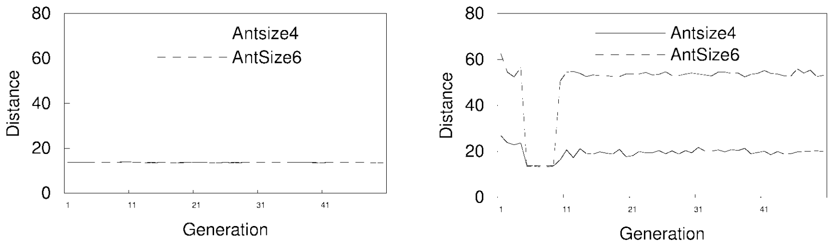
3.1 Statistical Results

What we are looking for in this section is some pattern of how the two sizes of ants organize themselves spatially. As can be noted in figure 3, there is a not surprising difference in the fitness obtained in the two test set-ups. The setup without spatial constraints outperforms the one with spatial constraints by a factor of 2,5 (a). From (b) we can tell that without spatial constraints the two sizes of ants will drop food items at the goal, whereas in the right graph the big ants settle on an average dropping distance that is somewhat near the nest border, and the small ants drop close to the centre. In this setup the optimal thus seems that the big ants drop the food further away from the centre. It does not pay off to travel the narrow path to the centre. The relative premature flattening of the fitness is probably due to our non-recombining crossover operator. A diversity measure in the start revealed that the population spanned some 60% of the search space, but only 1% in the end.

In table 4 and 5 different statistical numbers for the time spend, and the number of food items dropped is listed according to the zone. Table 4 confirms the plots in figure 3 in that the different ants spend an equal amount of time in the zones, as well as having an equal dropping ratio pattern in the setup without spatial constraints. With introduction of the spatial constraints we see a new pattern in table 5. In zone 0, the small ants dominate in time steps spend, and we can see in the food drops part of the



(a) Fitness



(b) Average dropping distance from nest

Fig. 3. Statistical results. The left column is results from the test setup without spatial constraints; whereas the right column is with spatial constraints.

table, that they have no less than 96% of the drops in this zone. This is actually quite odd, because if the big ants had carried food in this area they would have dropped it. The only explanation is that they have had a hard time getting out of the nest probably after dropping food further away. In zone 1 we see that the small ants still spend more time here than the big ants but they have only 25% of the food-droppings. This could only indicate that they forage much of their food here, and bring it to zone 0. Seemingly the supply of food around the nest is not abundant enough for the small ants to stay there, so they spend quite some time in zone 2 also. In zone 2 the big ants have most of their droppings and most of their time. In all we see a clear spatial pattern of where the different ants spend their time, and also where they deliver food, namely that the small ants spend most of the time inside the nest, bring food from the nest entrance (zone 1) to the nest centre. The ratio between big ants and small ants does not seem to be optimal though, or the distance from the perimeter to the nest entrance, and from the nest-entrance to the centre is not right. One would expect that the big ants would have the advantage in the obstacle-less zone 2 and moving faster also, whereas the small ants have an advantage with higher dexterity in zone 0 and 1, but they move slower. As the small ants go forage in zone 2, there must be a lack of big ants. In test set-ups with pure big ants, or pure small ants we didn't get as high fitness as the mix population, so this division of the labour in spatial areas must actually represent a better and more optimal solution.

Table 1. Time steps spend and food items dropped in the different concentric zones. Simulation without spatial constraints. Table should be read column wise.

	Time spend			Food drops		
	Zone 0	Zone 1	Zone 2	Zone 0	Zone 1	Zone 2
Antsize 4	49%	48%	50%	41%	46%	0%
Antsize 6	51%	52%	50%	59%	54%	0%

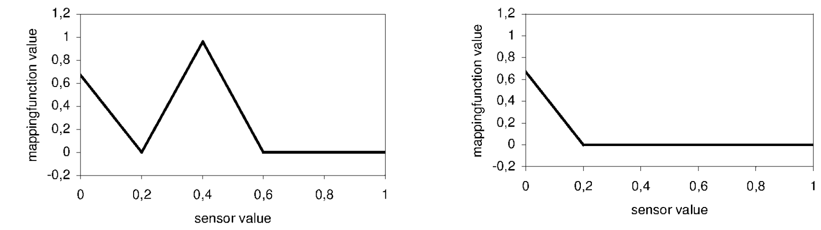
Table 2. Timesteps spend and fooditems dropped in the different concentric zones. Simulation with spatial constraints. Table should be read column wise.

	Time spend			Food drops		
	Zone 0	Zone 1	Zone 2	Zone 0	Zone 1	Zone 2
Antsize 4	63%	60%	40%	96%	25%	9%
Antsize 6	37%	40%	60%	4%	75%	91%

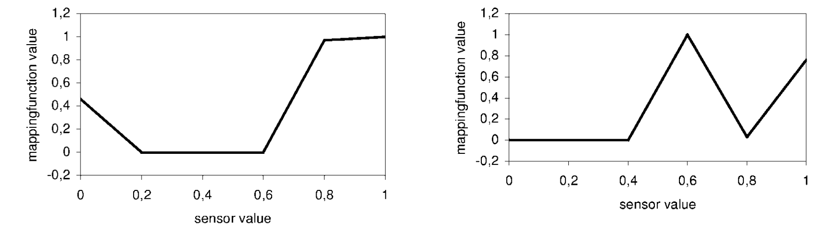
3.2 Evolved Mapping Functions

The mapping-functions that are encoded in the genes of the fittest colonies are listed in this section. Please note that the shape of one mapping function is co-evolved together with other mapping functions. So the mapping functions of all behaviours should be seen as a whole. In figure 4, we see that in both test set-ups, it is an important behaviour for the ants to find food (a) if it is not carrying any. The peak in the left column must be a stochastic flux of evolution as the sensor is binary-valued. Find-Home behaviour is important when the ant is carrying food (b). The takeFood behaviour has highest score at 0 in both set-ups. In the , which is when carry sensor is 0, and the foodsensor is 0. This means that the ant is not carrying food, and that food is 0 degrees just in front of ant and thus able to pickup. It seem from the ant sensor, that this behaviour in the non-spatial constrained set-up is most important when no ants are around (sensor value=0). The dropFood behaviour has optimum at 1 in both set-ups. In the spatially constrained setup the behaviour required values of 1 for all censors. So the ant should carry food, it should be very frustrated, sense a lot of food, and movement should be blocked by clay. This is a behaviour that could be coined kamikaze, in that only when all odds are against it, it will drop the food. (note the foodsensor is 1=360 degrees). Later we shall see a typical simulation snapshot where this is the case, namely at the edge of the nest. In the non-spatial constrained set-up the most notable difference being that the ant sensor need to have value 0, which indicate that crowding around the drop zone is not an issue in this set-up.

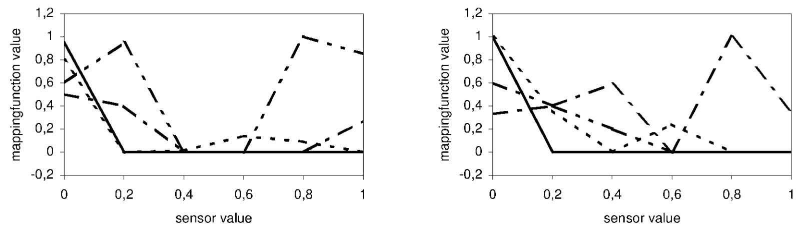
The emerged sensors seem quite reasonable. The dropping of food items has evolved to depend on high values of the ant and clay sensors, as well as high value for the frustration sensor. This means that the ants will not drop food prematurely in the outer zones. Dropping in zone 0 happens automatically, so this need not be handled by the behaviour. In terms of division of labour we see that the foraging task is split up when the spatial constraints are too high.



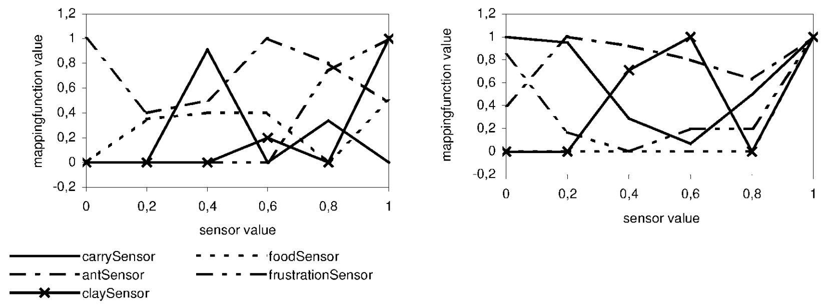
(a) findFood behaviour



(b) findHome behaviour



(c) takeFood behaviour



(d) dropFood behaviour

Fig. 4. The resulting mapping functions encoded in the fittest colonies. The functions in the left column is from simulations without spatial constraints, whereas the functions in the right column has spatial constraints. The legend in the bottom-left explains the lines. To calculate a score for a behaviour one would take the average of all mapping functions applied to their sensor input.

3.3 Emergent Patterns

As a typical view of a client simulation we see figure 5. We see roughly 3 times the number of small ants in the nest compared to big ants. Big ants do get into zone 0, as at least one food carrying big ant is heading for the nest centre, and another big one is leaving out bounds for more food. Note the food scattered around the nest boundary, which is ready for small ants to pickup. In the left part of the nest area not much food is placed around the nest, so here quite a few smaller ants have gone for food further out. This supports the observations in earlier sections on the 40% time used in zone 2 for the small ants. With this, relative low, number of ants in the world, we will primarily see emergent effects caused by the clay objects, as the ants are too few to really constrain each other spatially.

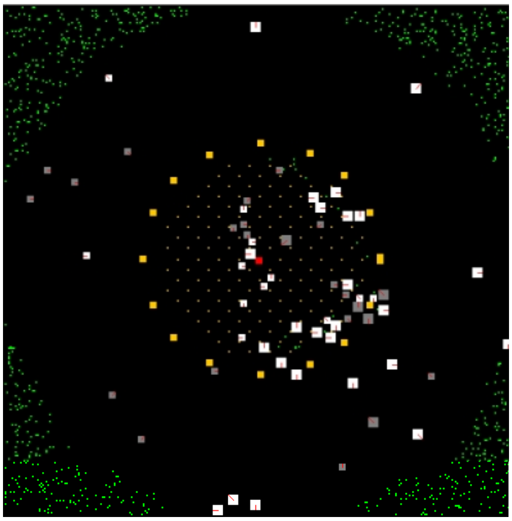


Fig. 5. The AntColonyEA Client with graphics. The small dots in the perimeter are food, whereas the square and small dots in the centre are clay blocks. The squares with a direction-vector drawn on them are ants. The darker ones carry food, the light ones carry no food.

4 Discussion

In the results sections we have seen good evidence for a spatially division of the ants performing a foraging task both in a non-constrained, and in a constrained environment. The difference between the two tests set-ups show that when spatial constraints are introduced the division of the labour changes, such that the individuals that are better fit for a specific subtask performs this. The model introduced, assumes very little, and yet evolves priority mapping-functions that encode a dynamic behaviour that divides the labour. So what was important in the model? For one, the ability of the ants to sense their immediate fit for the foregoing task at a specific location with a specific setup of objects in the vicinity seemed very important. Test simulations without the frustration sensor confirmed that the right behaviours were difficult to evolve without this introspective sensor. Many observations in nature justify this introspective-sensor, as e.g. bees estimate their own value at a specific task by measuring waiting-times. If they wait too long they will give up, and revert to other tasks. As the only difference between ants were their size, which in turn implied differences in their sensory inputs in certain situations, this sensor could trigger dynamic behaviour that depended on the specific ant. This idea seems to underlie many theories of division of labour. Namely that it is an emergent spatial effect that comes about by the interactions of the ants, the structure of the task and the ant's ability to transfer to subtasks if needed [6,7]. That the small ants eventually ended up in the nest area primarily is not something we coded, but this was an emergent effect of the big ants being quicker at fetching food, and the availability of food in the nest area for the small ants, otherwise they would transfer to foraging in the outer zones. It is also clear from our testing that the setup of the spatial constraints means a lot to the emergent patterns.

Implications for computer science could be in distributed autonomous software-agents. This study suggest that if these agents are equipped with measuring/sensory capabilities that enable them to estimate their own fit/success of a particular job, and then change behaviour or task accordingly to obtain better fit, then dynamic behaviour could be the result. The good thing about the ants in this study is that they are not smart by themselves, but they change behaviour based on own measurements, e.g. via frustration sensors, and yet the emergent patterns of cooperation at the overall level seems intelligent.

Acknowledgments. The author would like to thank Ilja Booij for cooperation on the simulation framework. Thiemo Krink for advice and guidance (and for generally being optimistic), Rasmus K. Ursem, Rene Thomsen and Peter Rickers for constructive advice.

References

1. Bonabeau, E. 1999. Swarm Intelligence , From Natural to Artificial Systems. Oxford University Press.
2. Bourke, A.F.G. & Franks, N.R. 1995. Chapter 12. Social evolution in Ants. Monographs In Behavior And Ecology. Princeton.
3. McDonald, P. & H. Topoff. 1985. Social regulation of behavioral development in the ant. Journal of Comparative Psychology 99: 3-14.
4. Wilson, E.O. 1985a. Between-caste aversion as a basis for division of labour in the ant *Pheidole pubiventris*.
5. Wilson, E.O. 1985b. The principles of caste evolution. Behavioral Ecology and Sociobiology 16: 89-98.
6. Krink T. 1999. Cooperation and Selfishness in Strategies for Resource Management. Marine Environmental Modeling Seminar,Lillehammer, Norway.
7. Tofts, C. & N.R. Franks. 1992. Doing the right thing: ants, honeybees and naked mole-rats. Trends in Ecology and Evolution 7:346-349.
8. Tofts, C. 1993. Algorithm for task allocation in ants (A study of temporal polyetism). Bulletin of Mathematical Biology 55:891-918.
9. ferber, J. 1999. Multi-agent Systems – an introduction to distributed artificial intelligence. Addison Wesley.