Learning Useful Communication Structures for Groups of Agents

Andreas Goebels

International Graduate School of Dynamic Intelligent Systems
Knowledge Based Systems, University of Paderborn, Germany
swarmgroup@upb.de

Abstract. Coordination of altruistic agents to solve optimization problems can be significantly enhanced when inter-agent communication is allowed. In this paper we present an evolutionary approach to learn optimal communication structures for groups of agents. The agents learn to solve the *Online Partitioning Problem*, but our ideas can easily be adapted to other problem fields. With our approach we can find the optimal communication partners for each agent in a static environment. In a dynamic environment we figure out a simple relation between each position of agents in space and the optimal number of communication partners. A concept for the establishment of relevant communication connections between certain agents will be shown whereby the space the agents are located in will be divided into several regions. These regions will be described mathematically. After a learning process the algorithm assigns an appropriate number of communication partners for every agent in an - arbitrary located - group.

1 Introduction

Multi Agent Systems (MAS) and Swarm Intelligence (SI) are two quite recent but very promising topics in current computer science research. SI deals with large sets of individuals or agents that can be seen as a self organizing system showing emergent behaviour[1][2]. Ideas from biology are used often and successfully to solve (optimization) problems in the computer science area. In both fields, communication between the single agents or particles plays an important role. In nature this communication is, for instance, realized with the environment as communication partner, the so called stigmergy concept, first introduced by the biologist Grassé [3], or with special dance moves that can be found at several bee colonies[4]. In both examples the concept of *locality* and *self organization* plays an important role. In most swarms, flocks or schools in nature we can hardly observe global communication.
If we have to solve an optimization problem and need inter-agent communication to enhance or even enable solutions, we could make use of a complete communication structure that allows direct communication between all pairs of agents. But if we are in settings that deal with a huge number of agents, such complete structures might produce high communication costs and/or are

not manageable because of information inferences or other real world problems. Therefore, it would be nice if we would have a concept that can produce a very small and cheap communication structure without significantly reducing the quality of the solution.
In this paper, we consider the *Online Partitioning Problem (OPP)* introduced in [5]. This problem, which is located in the area of Multi Agent Systems and Swarm Intelligence, deals with the association of agents with very limited and mostly local knowledge with different tasks represented as targets. The agents distribute themselves in an Euclidean space according to the following three objectives:

(1) The agents have to be distributed uniformly.
(2) Minimize the overall distance toward the targets.
(3) The abilities of the agents should be very simple.

Each of these goals is oppositional to any other, so we look for the best possible solution fitting in all objectives in quite an acceptable way. The knowledge of each agent is limited to a (preferable small) communication radius. They are able to communicate with their direct neighbours and know the distance to all targets according to their position. A more detailed description of the abilities of the agents can be found in [5]. There, several basic strategies have been presented to distribute a small number of agents onto two targets, coping with the three objectives mentioned before. It turns out that the communicative strategies perform better than the non-communicative ones. In general, Matarić discusses in [6] some advantages of using communication in multi agent systems to reduce locality by addressing two key problems, the hidden state and credit assignment problem.
In this paper we present an algorithm that is able to construct a successful communication structure to solve the *OPP* by defining useful communication connections between agents. This is done by dividing the space into regions depending on the position of each agent in relation to the targets and by learning an ideal number of communication partners for agents in these regions.
This paper is organized as follows. In the next section some terms will be defined. As an introduction, we roughly present an idea that deals with static settings. In the main section 4 we present our approach for dynamic groups of agents and do some mathematical considerations of the single regions we divided the space into. To show the quality of our approach we present in section 5 several simulation runs.

2 Definitions

In this paper, we denote the set of all agents with $\mathcal{A} = \{a_1, ..., a_n\}$ and the set of targets with $\mathcal{T} = \{t_1, ..., t_m\}$ $(n, m \in \mathbb{N})$. $\delta(p_1, p_2)$ defines the geometric distance between two points in the Euclidean space. This function works in the same way if we consider two arbitrary agents a and a' or an agent a and a

target t, then $\delta(a, a')$ and $\delta(a, t)$ calculate the distance between the positions of two agents or between an agent and a target, accordingly.

3 The static approach

In this section we consider a static setting, i.e. a set of agents on fixed positions in a two-dimensional, Euclidean space dealing with the *OPP*. They have to decide for one target regarding the objectives we mentioned in the introduction. For given parameters dictating communication costs and parametrizing the objective function for the OPP we try to find an optimal communication structure among the agents. Because of the unknown structure and the size of the solution space we make use of a genetic algorithm to search for good solutions. With this approach, a solution that optimally fulfils an evaluation function can be found very fast. We will give only a rough idea of this algorithm, more details and the results can be found in a master thesis [7] that was done under our supervision.

3.1 Evaluate the Quality of a Communication Structure

The quality or fitness f of a communication structure can be calculated at any time by the following formula. The notation is related to the notation in [7]. We sum up the single optimization criteria, i.e. the partitioning quality, the distance quality and the communication costs, and weight the single parts.

$$f = \alpha \cdot \left(\frac{\prod_{i=1}^{m} b_i}{\prod_{i=1}^{m} o_i} \right) + \beta \cdot \left(\frac{\sum_{i=1}^{n} \min_{j=1..m} (\delta(a_i, t_j))}{\sum_{i=1}^{n} \delta(a_i, target(a_i))} \right) + \gamma \cdot \left(1 - \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} c_{(i,j)} \cdot \delta(a_i, a_j)}{\sum_{i=1}^{n} \sum_{j=1}^{n} \delta(a_i, a_j)} \right)$$

$$\text{with } \alpha + \beta + \gamma = 1; \alpha, \beta, \gamma \geq 0$$

In this formula, b_i denotes the number of agents that have chosen the target t_i in the current partitioning decision and o_i the number of agents that would have chosen target t_i in an optimal partitioning. $target(a_i)$ defines the target currently chosen by agent a_i and $c(i, j)$ is either 1 or 0 depending on the existence of a directed communication link from agent a_i to a_j. The highest possible fitness is $f = 1.0$.

3.2 The genetic algorithm

We implemented a standard genetic algorithm and guide the search among all possible communication structures by the mentioned fitness function that consist of three summands representing the different objectives. Therefore, we consider a multi-objective optimization problem and its single objective representation.

One individual in our GA is a $n \times n$-matrix $\mathcal{C}$ describing the connectivity of the agents among each other. A '1' on position (i, j) allows agent a_i to communicate with agent a_j (directed communication). In other words, $\mathcal{C}$ is the adjacency matrix of the communication graph of the agents.
As a selection operator we use the *Best* selection and for mutation we simply swap bits in $\mathcal{C}$ with a low probability. The crossover method is a modification of the *Single-Point Crossover*. We apply this operator to two communication matrices $\mathcal{C}_1$ and $\mathcal{C}_2$ by choosing a random field (i, j) with $i, j \in [1; n]$ in the communication matrix. The two new individuals will exchange a corresponding rectangular part of the matrix defined by (i, j) as the upper left and (n, n) as the lower right corner.

4 The Dynamic Approach

In the former section we introduced a static approach to learn optimal communication structures for a given set of fixed agents. But this structure strongly depends on the special setting it was trained on and cannot be used for other groups of agents or other positions of the same agents, especially if we consider communication costs that are constrained by the distance between two communication partners. In this section, we will present an idea to learn a useful communication structure that is independent from the distribution of the agents in space.
Therefore, we construct a communication network for a dynamic setting in a totally different way. The agents do not learn what the best communication partners are, but they try to find out how many communication partners are useful for the position they are located at (depending on the position of the targets). That means, the agents learn a function that connects a region that can be computed locally with an ideal number of communication partners. Since the agents do not know the extension of the simulation area, the agents have to calculate the area they are in only with regard to the distances to the targets. Therefore, we calculate a q-value for each agent position (x, y) by

$$q_{x,y} = \frac{min(\delta((x,y),t_1),...,\delta((x,y),t_n))}{max(\delta((x,y),t_1),...,\delta((x,y),t_n))} \in \mathbb{R}$$

Or, to put it in a more informal description, we calculate the quotient for each agent position by dividing the distance to the closest target by the distance to the farthest target[1]. With this procedure we can calculate values that represent the area an agent is in without paying attention to its real distance. In the further text we will call this value the q-value. Because we are in a continuous space, there is an infinite number of different q-values. Therefore we combine the different q-values in intervals of the same size. For l categories, we obtain the following intervals $I_1, ..., I_l$ with

[1] In this paper we focus on settings with two targets. If we would consider a higher number, we maybe will have to make the calculation of the q-value more complicated. This will be focus of an upcoming paper.

$$I_k = \begin{cases} \left[\frac{k-1}{l}; \frac{k}{l}\right] & for\ \ k \neq l \\ \left[\frac{k-1}{l}; 1\right] & for\ \ k = l \end{cases}$$

4.1 Our Approach

We solved the *Online Partitioning Problem (OPP)* for huge agent sets and enhanced the knowledge base of a selection of agents by enabling communication with its neighbours.

Table 1. The number of communication partners based on the q-interval. For each interval, an appropriate number of communication partners can be defined.

q-interval	I_1	I_2	...	I_l
number of communication partners	n_1	n_2	...	n_l

Therefore, the agents have to learn the number of communication partners depending on the q-interval they are located in. Or, in other words, they learned an appropriate assignment for each n_i in table 1. We call such a table q-table.

4.2 Size and Properties of the q-Intervals in Space

In this section we will give a short mathematical insight into the regions we created with our intervals. We consider an arbitrary q-value, denoted by q'. All positions in space that produce exactly the value q' are located on two circles with the same radius r around two centre points. The targets are somewhere inside this circles. The distance between the two targets is fixed and denoted by D.

Theorem

All points in space that have one specific q-value according to two targets t_1 and t_2 on positions (t_{1_x}, t_{1_y}) and (t_{2_x}, t_{2_y}) lie on the circles C_1 and C_2 with centre points

$$M_1 = \left(\left(\frac{t_{2_x} - q^2 \cdot t_{1_x}}{1-q^2}\right), \left(\frac{t_{2_y} - q^2 \cdot t_{1_y}}{1-q^2}\right)\right), M_2 = \left(\left(\frac{t_{1_x} - q^2 \cdot t_{2_x}}{1-q^2}\right), \left(\frac{t_{1_y} - q^2 \cdot t_{2_y}}{1-q^2}\right)\right)$$

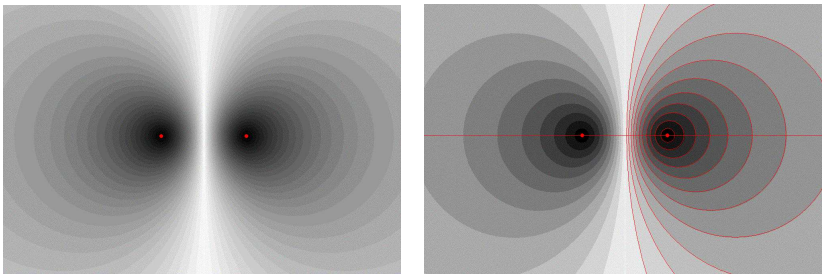
and radius $r = \frac{q \cdot D}{(1-q^2)}$.

Proof

q is calculated for an arbitrary point $p = (x, y)$ by the formula

$$\frac{dist.\ to\ nearest\ target}{dist.\ to\ farthest\ target}.$$

Without loss of generality we assume that target t_2 is the nearest one and t_1 the farthest one. Therefore, q can be expressed by:



(a) (b)

Fig. 1. Figure (a) shows the distribution of the q-values when calculated for arbitrary positions in space. Each grey tone represents one specific q-interval. In this example there are 30 different q-intervals. In (b), the circles for the interval borders (here we have 12 intervals), obtained by our mathematical examination, are visualized. They perfectly cover the regions.

$$q = \frac{\sqrt{(x - t_{2_x})^2 + (y - t_{2_y})^2}}{\sqrt{(x - t_{1_x})^2 + (y - t_{1_y})^2}}$$

This can be transformed to:

$$\left(\sqrt{\frac{q^2 \cdot t_{1_x}^2 + q^2 \cdot t_{1_y}^2 - t_{2_x}^2 - t_{2_y}^2}{1 - q^2} + \left(\frac{t_{2_x} - q^2 \cdot t_{1_x}}{1 - q^2} \right)^2 + \left(\frac{t_{2_y} - q^2 \cdot t_{1_y}}{1 - q^2} \right)^2} \right)^2$$
$$= \left(x - \left(\frac{t_{2_x} - q^2 \cdot t_{1_x}}{1 - q^2} \right) \right)^2 + \left(y - \left(\frac{t_{2_y} - q^2 \cdot t_{1_y}}{1 - q^2} \right) \right)^2$$

and this can be simplified to a standard circle equation:

$$\left(\frac{q \cdot D}{(1 - q^2)} \right)^2 = \left(x - \left(\frac{t_{2_x} - q^2 \cdot t_{1_x}}{1 - q^2} \right) \right)^2 + \left(y - \left(\frac{t_{2_y} - q^2 \cdot t_{1_y}}{1 - q^2} \right) \right)^2$$

These are centre point M_1 and radius r in our Theorem. If target t_1 is the nearest one, we obtain the other centre point M_2 by using the same transformations. □

4.3 The Genetic Algorithm

There is a huge number of possible assignments for such a structure as presented in table 1, especially when dealing with large numbers of agents. Each n_i can be assigned to a value from $\{0, ..., (n-1)\}$ with $n = |\mathcal{A}|$ representing the number of agents. Therefore, we have $(n-1)^l$ possible assignments. We use a genetic algorithm to search for good ones because we have no prior information about the structure of the solution space. The fitness of a solution obtained with a table assignment is the quality of the solution of the *OPP* . It is set in relation

to the optimal solution, regarding communication costs, and is calculated by the formula:

$$fitness(tableAssignment) = \alpha \cdot fitness_{OPP} + (1 - \alpha) \cdot fitness_{Communication}$$

The fitness of a OPP solution is calculated as in [8] by

$$fitness_{OPP} = \beta \cdot \left(\frac{\prod_{i=1}^{m} b_i}{\prod_{i=1}^{m} o_i} \right) + (1 - \beta) \cdot \left(\frac{\sum_{i=1}^{n} \min_{j=1..m} (\delta(a_i, t_j))}{\sum_{i=1}^{n} \delta(a_i, target(a_i))} \right)$$

We use the same notation as we did in section 3.

The **Communication Costs** in the fitness function will be calculated by regarding the communication distances between two agents that are allowed to communicate. The number of communication partners is defined in the q-table. When establishing n_k connections for an agent located in the interval I_k, this agent will create communication lines to its n_k nearest agents. The sum of these costs will then be set in relation to the maximum costs for communication that could appear if it holds for each entry in the q-table that n_i is equal to $(n-1)$. Hence, we can define a partial fitness function for the communication costs:

$$fitness_{Communication} = \frac{q - table\ defined\ communication\ graph\ costs}{complete\ communication\ graph\ costs}$$

We assume that a subset of agents that own communication connections among each other will be able to calculate an optimal partial solution for the *OPP*.

The **crossover** operator is simple, we use *One-Point Crossover*. Therefore, a random point p from $\{1, ..., l\}$ is chosen. Then we create two new q-tables by recombining the tables from two parental individuals split at this particular point (or column) p. We think that we can maintain coherences between the table entries with this operator if they exist.

For the **selection** of individuals for the next generation we implemented the *Roulette Wheel* and the *Best* selection. In comparison runs the *Best* selection shows slightly better results, therefore we made our final experiments with this method. For both algorithms, we use a (μ, λ)-scheme and chose 50% of the individuals for the new generation out of the old generation.

For the **mutation** of a q-table we make use of two mutation parameters, p_I defines the probability for mutating one individual. The second parameter p_t determines the probability of mutating one table entry. When an entry I_k has been selected to become mutated, we adjust the n_k value in the table by adding a random value $r \in \{-(n-1), ..., (n-1)\}$ to the entry and check if the value is out of range with the formula $mutation(n_k) = max(min(n-1, n_k + r), 0)$.

4.4 Our Algorithm

We use a genetic algorithm to find the appropriate number of communication partners for each interval in the q-table. The most interesting part of the algorithm is the calculation of the fitness of an individual in the population, i.e. the fitness of a q-table. To rate such a q-table $\mathcal{Q}$, we test the quality of a set of

agents working on the *Online Partitioning Problem* that use a communication structure developed from $\mathcal{Q}$. The fitness of each $\mathcal{Q}$ can be calculated by the following algorithm:

```
001:  FUNCTION double calculateFitness(qTable Q)
002:  {
003:    agentSet = new random set of agents;
004:    place targets on random positions in space;
005:    agentSet.createCommunicationGraph(Q);
006:
007:    commFitness = calculateCommunicationFitness(agentSet);
008:    oppFitness = calculateOPPSolutionFitness(agentSet);
009:
010:    RETURN α · oppFitness + (1 − α) · commFitness;
011:  }
```

The function $calculateCommunicationFitness(agentSet)$ simply applies the communication fitness function as described in 4.3. The higher this value is the less communication is used.[2] The more interesting function is the one calculating the *OPP* solution fitness, this is shown here in more details:

```
001:  FUNCTION double calculateCommunicationFitness(agentSet)
002:  {
003:    // create reference solution
003:    d = minimal overall distance to targets in optimal partitioning;
004:    s = optimal number of agents on each target in optimal partitioning;
005:
006:    FOR EACH agent a in agentSet DO
007:    {
008:      IF (#outgoingConnections(a) > 0) //derived from q-table
009:        calculateLocalOptimalSolution(a, communicationPartners);
010:      ELSE
011:        choose nearest target;
012:    }
013:    d' = calculateDistanceFitness(agentSet, d);
014:    s' = calculateDistributionFitness(agentSet, s);
015:
016:    RETURN β · d' + (1 − β) · s';
017:  }
```

The function $calculateLocalOptimalSolution(a, communicationPartners)$ in line 9 assumes that an agent can calculate the optimal partitioning for the agents it communicates with. This is quite an idealistic picture because we still have a hard problem, but we can take this calculation power into account by increasing the influence of the communication costs for the fitness function. Anyway, if this function can calculate only an approximation, our algorithm will still work.

The pseudocode algorithms show only the most important steps of our algorithm, for a more detailed insight you can have a look at the original Java sources that are available for download and further experiments via our webpage[3].

[2] In our simulations we repeated lines 3-8 several (five) times to obtain more meaningful fitness values.

[3] http://www.upb.de/cs/ag-klbue/de/staff/agoebels/index.html

5 Results

In this paper, we concentrate on the results for the dynamic approach presented in section 4, the results for the static approach (section 3) can be found in [7]. There, a good communication matrix could be found fast for every given fixed set of agents.

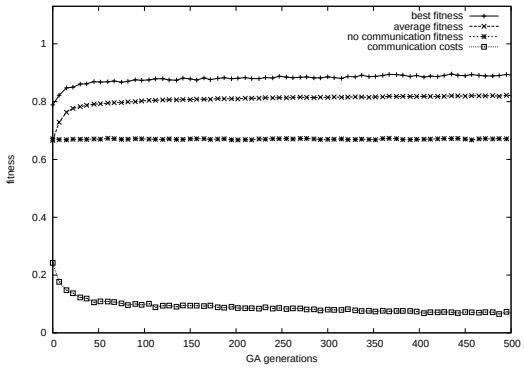


Fig. 2. This figure presents the development of the fitness over 500 generations. We show the average fitness of the whole generation and the fitness development of the best individual in population. This graph illustrates the average result over 25 runs. The parameters we made use of can be found in the source code package. The fitness rises while the communication cost could be reduced. The single graphs are smoothed with a Bezier curve for better visibility.

5.1 Fitness Development

First of all, we examined how the fitness of the GA develops. Figure 2 shows a typical fitness development. Both the best and the average fitness rise very fast to a high level and remain there. As a reference we present the fitness value of a non communicative algorithm choosing always the nearest target for each agent. This reference fitness is significantly lower than the fitness value achieved with our approach. By adjusting the weights for the communication costs we can obtain any fitness value between 1.0 (no communication cost, $\alpha = 1$) and the reference function ($\alpha = 0$). Hence, we can conclude that inter-agent communication enhances the solution quality of the whole group and our approach finds good *OPP* solutions for given communication costs or restrictions.

5.2 Development of q-Table values

Once we could see that our idea works, we wanted to get more insight into the communication structure the agents learn. Therefore, we observed the changes

of the q-table entries during the learning process. Figure 3 shows a typical picture representing the value development.

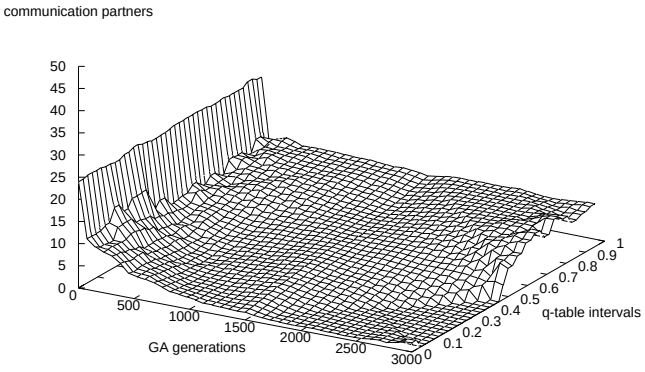


Fig. 3. The development of the q-table entries during the learning process.

After several hundred generations the q-tables look similar for all settings. In this figure the average over all q-tables in 25 runs is presented. In the early generations the number of communication partners is high and nearly identical for all intervals. But the communication structure becomes fastly sparser and in the last generations we can see that the q-table can be divided into 2 parts. For intervals containing small q-values the agents learned to have nearly no communication partners[4]. For q-values greater than 0.5 it seems to make sense to communicate with a small number of neighbours to increase the fitness to a near optimum value. One other key result is that the overall number of communication connections between agents is very low compared to the maximum possible value. This shows that a communication structure does not necessarily have to be very complex or massive if we generate it in an intelligent way.

6 Conclusion & Future Work

In this paper we presented a genetically guided approach to learn qualified communication structures for sets of agents to solve an optimization problem. For a

[4] The intervals I_1 and I_2 contain all q-values below 0.1. As we saw in 4.2, the area in space representing all possible values in these intervals is very small compared to the remaining space, hence the probability for an agent to be placed in one of these areas is very low and it does not influence the fitness significantly.

static environment we presented a rough idea how the optimal communication structure can be found by an algorithm adjusting a communication matrix. For dynamic and random settings we presented a new approach which offers guidelines to create a small set of communication connections. Therefore, only the position in space in relation to some targets is necessary, the optimal or near optimum number of communication partners can be found by our approach.
In our future research we will restrict the values in the q-table and will try to learn or cope with restricted communication distances. This will enlarge the possible number of real-world applications. We also will examine the influence of the different probabilities for each q-interval on the fitness development and the overall solution quality.

References

1. Eric Bonabeau, Marco Dorigo, and Guy Theraulaz, *Swarm Intelligence - From natural to artificial Systems*, Oxford University Press, 1999
2. James Kennedy and Russel C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, 2001
3. P.-P. Grassé, *La Reconstruction du nid et les Coordinations Inter-Individuelles chez Bellicositermes Natalensis et Cubitermes sp. La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs*, Insectes Sociaux, 1959.
4. K. von Frisch, *The Dance Language and Orientation of Bees*, Harvard University Press, Cambridge, 1967
5. Andreas Goebels, Hans Kleine Büning, Steffen Priesterjahn, Alexander Weimer, *Towards Online Partitioning of Agent Sets based on Local Information*, Proceedings of the International Conference on Parallel and Distributed Computing and Networks (PDCN), 2005
6. Maja J. Matarić, *Using Communication to Reduce Locality in Distributed Multi-Agent Learning*, Journal of Experimental and Theoretical Artificial Intelligence, special issue on Learning in DAI Systems, Gerhard Weiss, ed., 10(3), 1998, 357-369.
7. Lars Beckmann, *Evolutionäre Entwicklung und Optimierung von Kommunikationsstrukturen zur Koordination von Agenten [Evolutionary Development and Optimization of Communication Structures for Agent Coordination]*, Master Thesis, Univ. of Paderborn, 2005
8. A. Goebels, A. Weimer, S. Priesterjahn, *Using Cellular Automata with Evolutionary Learned Rules to Solve the Online Partitioning Problem*, in Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05), Edinburgh, 2005, pp. 837-843