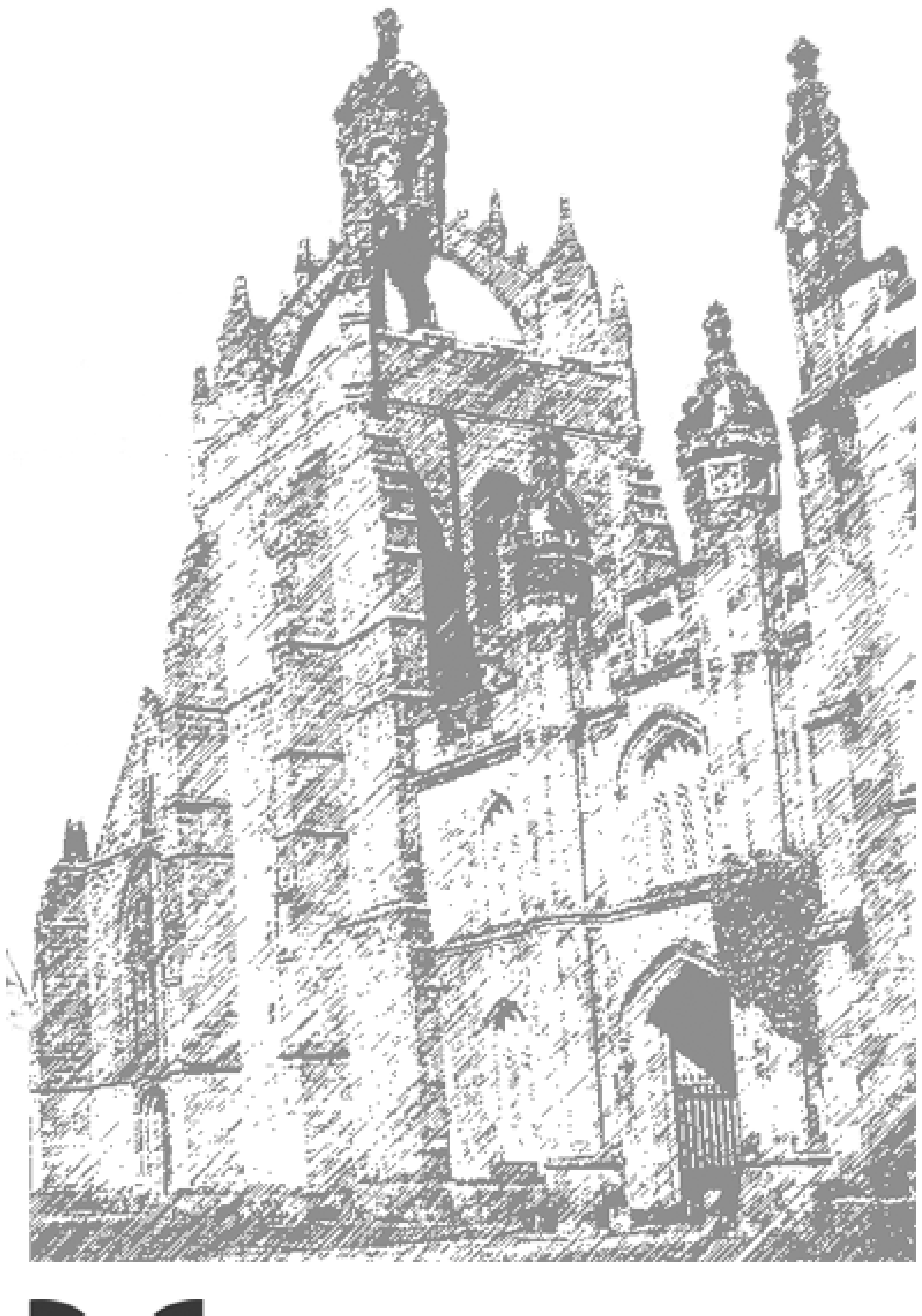
Swarm Intelligence Algorithms and Applications

AISB '08

AISB 2008 Proceedings Volume 11



GLoRiClass

UNIVERSITY OF ABERDEEN

AISB 2008 Convention Communication, Interaction and Social Intelligence

1st-4th April 2008

University of Aberdeen

Volume 11:
Proceedings of the
AISB 2008 Symposium on Swarm Intelligence Algorithms and Applications

Published by
The Society for the Study of Artificial Intelligence and Simulation of Behaviour

http://www.aisb.org.uk/convention/aisb08/

ISBN 1 902956 70 2

Contents

The AISB'08 Convention ii
Frank Guerin & Wamberto Vasconcelos

Symposium Preface iii
Aladdin Ayesh

Visualizing Bacteria Quorum Sensing 1
Maria Schwarz, Daniela Romano & Marian Gheorghe

Trained Particle Swarm Optimization for Ad-Hoc Collaborative Computing Networks 7
Shahin Gheitanchi, Falah Ali & Elias Stipidis

Controller-Agent based approach for Solving Distributed Constraint Problem 12
Sami Al-Maqari & Habib Abdularb

Introducing Bar Systems: A Class of Swarm Intelligence Optimization Algorithms 18
Esteve del Acebo & Josep Lluis de la Rosa

A Comparison between GAs and PSO in Training ANN to Model the TE Chemical Process Reactor 24
Malik Braik, Alaa Sheta & Amani Arieqat

Real Time Movement Cooredination Technique Based on Flocking Behaviour for Multiple Mobile Robots System ... 31
Ghada AlHudhud & Aladdin Ayesh

Aqua Swarms: Design and Implementation of Water Surface AUV 38
Mustafa Ozkan Daglioz & Aladdin Ayesh

Ant Colony Optimisation for Large-Scale Water Distribution Network Optimisation 44
Laura Baker, Ed Keedwell & Mark Randall-Smith

Estimation of Hidden Markov Models Parameters using Differential Evolution 51
Ângela A. R. Sá, Adriano O. Andrade, Alcimar B. Soares & Slawomir J. Nasuto

Exploration vs. Exploitation in Differential Evolution 57
Ângela A. R. Sá, Adriano O. Andrade , Alcimar B. Soares & Slawomir J. Nasuto

Toward a Unified Framework for Swarm Based Image Analysis 64
Walther Fledelius & Brian Mayoh

The AISB’08 Convention: Communication, Interaction and Social Intelligence

As the field of Artificial Intelligence matures, AI systems begin to take their place in human society as our helpers. Thus it becomes essential for AI systems to have sophisticated social abilities, to communicate and interact. Some systems support us in our activities, while others take on tasks on our behalf. For those systems directly supporting human activities, advances in human-computer interaction become crucial. The bottleneck in such systems is often not the ability to find and process information; the bottleneck is often the inability to have natural (human) communication between computer and user. Clearly such AI research can benefit greatly from interaction with other disciplines such as linguistics and psychology. For those systems to which we delegate tasks: they become our electronic counterparts, or agents, and they need to communicate with the delegates of other humans (or organisations) to complete their tasks. Thus research on the social abilities of agents becomes central, and to this end multi-agent systems have had to borrow concepts from human societies. This interdisciplinary work borrows results from areas such as sociology and legal systems. An exciting recent development is the use of AI techniques to support and shed new light on interactions in human social networks, thus supporting effective collaboration in human societies. The research then has come full circle: techniques which were inspired by human abilities, with the original aim of enhancing AI, are now being applied to enhance those human abilities themselves. All of this underscores the importance of communication, interaction and social intelligence in current Artificial Intelligence and Cognitive Science research.

In addition to providing a home for state-of-the-art research in specialist areas, the convention also aimed to provide a fertile ground for new collaborations to be forged between complementary areas. Furthermore the 2008 Convention encouraged contributions that were not directly related to the theme, notable examples being the symposia on “Swarm Intelligence” and “Computing and Philosophy”.

The invited speakers were chosen to fit with the major themes being represented in the symposia, and also to give a cross-disciplinary flavour to the event; thus speakers with Cognitive Science interests were chosen, rather than those with purely Computer Science interests. Prof. Jon Oberlander represented the themes of affective language, and multimodal communication; Prof. Rosaria Conte represented the themes of social interaction in agent systems, including behaviour regulation and emergence; Prof. Justine Cassell represented the themes of multimodal communication and embodied agents; Prof. Luciano Floridi represented the philosophical themes, in particular the impact of society. In addition there were many renowned international speakers invited to the individual symposia and workshops. Finally the public lecture was chosen to fit the broad theme of the convention – addressing the challenges of developing AI systems that could take their place in human society (Prof. Aaron Sloman) and the possible implications for humanity (Prof. Luciano Floridi).

The organisers would like to thank the University of Aberdeen for supporting the event. Special thanks are also due to the volunteers from Aberdeen University who did substantial additional local organising: Graeme Ritchie, Judith Masthoff, Joey Lam, and the student volunteers. Our sincerest thanks also go out to the symposium chairs and committees, without whose hard work and careful cooperation there could have been no Convention. Finally, and by no means least, we would like to thank the authors of the contributed papers – we sincerely hope they get value from the event.

Frank Guerin & Wamberto Vasconcelos

The AISB'08 Symposium on Swarm Intelligence Algorithms and Applications

The increasing complexity of the current world can be observed each day. Sustainable development for example consists of economical and social systems management within natural environment. The understanding of the whole leads to what we call territorial intelligence. This is an example of the interaction between living entities, e.g. humans, space and environmental elements, e.g. services. Within the space and governed by the elements, the living entities swarm leading to formation of communities and patterns exhibiting complex system dynamics.

The way of modelling these complex systems is often based on interactive networks dealing with the interconnection between all of the system components. The components themselves may be simple as separate individuals and exhibiting uninteresting behavior. However, the connection and interaction of these components make the simple collective behaviors to emerge into a complex often difficult to mathematically model behavior. This complex behavior is the result of the dynamics exhibited by the collective system.

Decision making in the emerging complex world of a collective need tools that are able to detect and manage emergent organizations through these networks. Distributed Artificial Intelligence (DAI) is the adapted conceptual trend which allows the proposal of some relevant solutions by relying on social and physical sciences models exhibited and observed in nature (e.g. ant colonies, molecular crystallisation, etc.).

In this search and management of emerging organization, swarm intelligence algorithms proved to be popular and effective methods to use. On the technological front, the increasing number of robotic systems, advances in nano technology, and the sheer complexity of modern enterprise systems, especially those boosting high degree of autonomy, makes the development of swarm intelligence timely and needed. These complex systems and the dynamics of a swarm collective can be easily related. At the same time, swarm intelligence algorithms require less computational powers than their equivalence traditional complex system.

Until now, swarm intelligence is often algorithmic approach than theoretically analyzed. However, it proved successful with a variety of applications such as optimization, image processing and simulation. Ant colony optimization and Particle Swarm Optimization algorithms are most popular algorithms in their area. They often extend to be applied in social simulation and image processing. Flocking algorithms are widely used in a variety of robot and reactive agent based applications.

This symposium brings papers in a variety of application areas covering wide range of swarm intelligence algorithms. Theoretical investigation is also represented although lightly.

Aladdin Ayesh

Visualizing Bacteria Quorum Sensing

Maria Schwarz, **Daniela Romano** and **Marian Gheorghe**[1]

Abstract. Large populations of bacteria communicate by sending into the environment some specific signalling molecules. A bacterium will sense the population density by the concentration of the signalling molecule. This process, known as "quorum sensing", is used in this paper to show the emergent behaviour of a bacterium colony in various circumstances. The biochemistry associated with quorum sensing has been modelled and an agent based approach has been used to simulate the behaviour of the colony. In addition, a 3-D environment has been created to illustrate how the emergent behaviour occurs within the colony as a result of local immediate real-time interactions, while a data file is generated to visualise the behaviour of the colony over time as a 2D graph.

1 INTRODUCTION

There are numerous ways in which members of various communities communicate. Even simple organisms have different ways to pass information among them. Quorum sensing (QS) is one of these communication mechanisms, which has been known since the end of the 1960s. A review of various bacterium populations, the LuxR-LuxI family, and how they communicate with each other is presented in [1].

Bacteria use QS to coordinate different behaviours. For example the light emission in luminescent bacteria (the *Vibrio fischeri* or *Vibrio harveyi*), division in *Escherichia coli*, or biofilm formation in infective bacteria like *Pseudomonas aeruginosa*. The most important role in QS is played by signalling molecules, called autoinducers, which are produced by bacteria. These autoinducers diffuse through the bacterium membrane into environment. The accumulation of it in the environment takes place if there is a high concentration of bacteria in that space. If this occurs, a special gene will be activated and an increase in signalling molecules production is observed [2]. Therefore the behaviour of the bacteria change and the population density will be controlled [1], [3].

The QS process is widely studied today due to its importance in regulating the colony behaviour, but also as a computational paradigm applied to investigate computer networks, artificial life phenomena [4].

The QS mechanism represents one of the most important cell-to-cell communication processes [4], which requires fine grain knowledge of the complex (micro-)biological system involved and the associated communication procedures in place. In this respect various approaches have be employed, ranging from mathematical models that address a global behaviour of the community to different agent based methods that represent every bacterium with its internal processes and local interactions between the community members. The agents, representing individuals in their environment, act in a totally autonomous manner [5] and coordinate their behaviour based on special rules. Thus, they interact with the other agents from their neighbourhood.

In order to study the behaviour of these colonies or to investigate certain properties of the computational paradigms associated with, adequate modelling and simulation tools have been provided. In the last few years improved computer technology has paved the way to visualise this process without major performance difficulties and it is possible to visualise a QS behaviour-using species of bacteria in three dimensions.

There have been built various agent based tools and software platforms. An example, which uses the agent-based method, is the visualisation of the motion and behaviour of a flock of birds [6]. Thereby each bird acts autonomously "to its local perception of the dynamic environment, the laws of simulated physics that rule its motion and a set of behaviours programmed into it by the 'animator'" [6].

There are also open source environments allowing to create agent-based models and to simulate them on various platforms. NetLogo and SWARM are both agent-based modelling environments suitable to specify, study and simulate complex systems with a high number of agents [7], [8]. Regrettably, both frameworks have a pretty elementary construction so that there is no parallel processing available. This problem is addressed by another agent based toolkit, called *MASON*, which is written in java [9]. This modelling toolkit provides parallel processing, which is controlled by a scheduler. Flexible Agent Modelling Environment (*FLAME*, for short) has been developed by a team of the University of Sheffield [10], and has a great flexibility in its usage and has been extensively employed to model complex biological and economical systems [10]. Every agent is abstractly represented like a state machine, all the processes occurring inside are specified in an XML format and they share a common special memory component that allows agents to communicate and store information [5].

The paper aims to uncover aspects related to the emergent behaviour of a colony of Vibrio fischeri bacteria with respect to various parameters, like cell density, amount of signals produced, signal diffusion in the environment. The following topics are covered. Section 2 gives a brief description of the model used in the agent-based approach. In section 3 an overview of the implementation of the agent model within FLAME framework and the visualisation part are presented. Section 4 presents the experiments concerning the simulation and shows the results obtained and their significance for the

[1] Department of Computer Science, The University of Sheffield, Regent Court, Portobello Street, Sheffield S1 4DP, UK
Email: `shef@schwarz-maria.de,`
`{d.romano, m.gheorghe}@dcs.shef.ac.uk`

emergent behaviour. The last section evaluates the findings and proposes updates to the existing approach.

2 THE CONCEPTUAL MODEL

Many species of bacteria regulate gene expression in response to increasing cell population density. Bacteria produce and release signalling molecules (called autoinducers) that accumulate in the environment as the cell density increases. When a certain threshold is achieved a signalling transduction cascade is triggered and leads finally to a change in behaviour by increasing the production of the signalling molecules and leading to other individual changes. In Vibrio fischeri population a protein called LuxI is responsible to synthesise an acylated homoserine lactone signalling molecule (HSL, for short) [3]. The HSL signalling molecule diffuses out of the bacterium, into the environment, following a gradient concentration or it penetrates in when the environment surrounding it has a higher concentration of these molecules than the bacterium. If the concentration of this autoinducer reaches a special threshold, and therefore there are a high number of bacteria in a small space, a protein called LuxR reacts with the autoinducers by producing a complex, called LuxR-HSL. This complex is responsible for the activation of the transcription of a specific gene and the bacterium enters a new state, it becomes quorated and starts emitting light [3].

The agent model consists of two agent types: bacterium-agent and environment-agent. The bacterium-agent model is defined according to the rules defined by Figure 1. Usually each bacterium produces signalling molecules and proteins LuxR at a low rate (rules r1 and r2, respectively). The signalling molecules freely diffuse into the environment (r9) and both, signalling molecules and proteins degrade in time (r10, r11). Due to the low rate production of autoinducers, diffusion and degradation, the number of signalling molecules in the bacterium is too low and it will not start becoming quorated and producing light.

However, in specific circumstances when a certain threshold is reached somewhere in a specific part of the environment, near some bacteria, which only happens if a high number of bacteria are in one small space, the signalling molecules will diffuse back into those bacteria and increase more the concentration of autoinducers within them, triggering the quoration mechanism which leads eventually to producing light. This process is described in our model by the production of the complex LuxR-signal (r3), which in high concentration of the signalling molecule will not decompose back into its components and will next bind to a certain gene (r5) and will remain there and increase the production of signalling molecule and protein LuxR (r7, r8).

$$r1 : [LuxBox] \xrightarrow{k1} [LuxBox, signal]$$
$$r2 : [LuxBox] \xrightarrow{k2} [LuxBox, LuxR]$$
$$r3 : [LuxR, signal] \xrightarrow{k3} [LuxR - signal]$$
$$r4 : [LuxR - signal] \xrightarrow{k4} [LuxR, signal]$$
$$r5 : [LuxBox, LuxR - signal] \xrightarrow{k5} [LuxBox - LuxR - signal]$$
$$r6 : [LuxBox - LuxR - signal] \xrightarrow{k6} [LuxBox, LuxR - signal]$$
$$r7 : [LuxBox - LuxR - signal] \xrightarrow{k7} [LuxBox - LuxR - signal, signal]$$
$$r8 : [LuxBox - LuxR - signal] \xrightarrow{k8} [LuxBox - LuxR - signal, LuxR]$$
$$r9 : [signal] \xrightarrow{k9} signal[]$$
$$r10 : [signal] \xrightarrow{k10} []$$
$$r11 : [LuxR] \xrightarrow{k11} []$$

Figure 1. Equations describing all chemical interactions [11]

The bacterium-agent described above contains five different molecules that play an important role in producing the signalling molecule. The process of production, diffusion, degradation, combination of these chemical molecules is controlled by an exact stochastic method developed according to Gillespie algorithm [12]. Usually this algorithm iteratively selects the next rule to be applied and computes the time for this to be performed. In order to do this the current concentration of the chemicals involved in a rule and the kinetic rate (k1 to k11, in Figure 1) are considered.

Each environment field, modelled as an environment-agent, contains a defined number of signalling molecules (at the beginning zero). This number of signalling molecules is changed by the movement of them according to the concentration gradient.

3 IMPLEMENTATION

The general architecture of the implementation is given in Figure 2 and contains the agent-based system implementation and the graphical interface part.

The upper part of Figure 2 refers to the agent framework FLAME [13], [14] and the bottom part represents the user interface to the system.

To implement FLAME we had to build a XMML Model, which is written as an xml file. It defines all the agents, their status, associated functions (activities) and their dependencies with respect to messages sent between agents; messages owned by each agent are defined. Additionally, a C-based file, where all the functions and their features are set, needs to be provided. This file depends on a xml model defining the structure of the state machine. The initial state of the system must be defined (for example how many agents are in the system, their initial distribution, the initial concentrations of the chemicals present in each bacterium etc.).

Instances of the bacterium-agents and of the environment-agent are immersed into FLAME. Every bacterium-agent will run a slightly modified version of Gillespie algorithm to select the next rule to be used, adapted to match the requirements of the agent framework. The initial concentration of the chemical molecules of the bacterium-agents will depend on various experiments made. The current Gillespie algorithm implementation computes only the next rule to be applied. All the agents of the system are able to send messages between them through a message board.

Thus the agents know independent of the other partners in the whole system what goes on in their close environment. Therefore every agent can behave irrespective of the size of the system or the collective number of agents [13], [14]. The only restriction is the dependency on the messages sent by agents in their close neighbourhood.

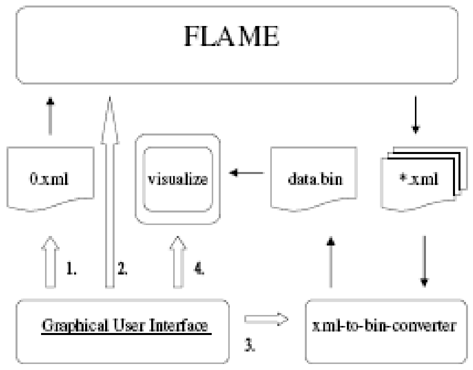
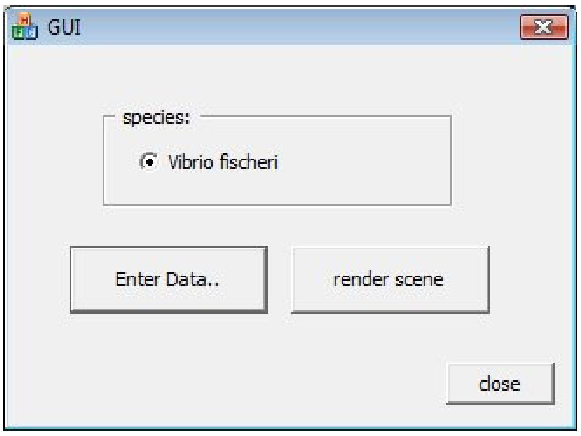
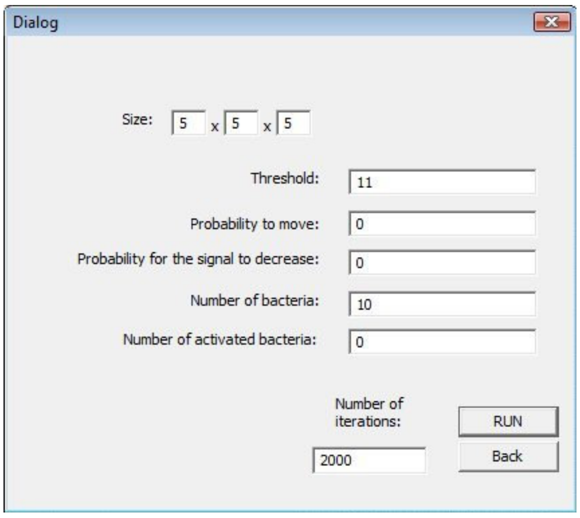


Figure 2. Overview of the implementation architecture




Dialog

Size: 5 x 5 x 5

Threshold: 11

Probability to move: 0

Probability for the signal to decrease: 0

Number of bacteria: 10

Number of activated bacteria: 0

Number of iterations: 2000

Figure 3. Screenshot of the graphical user interface and the child window of it

The Graphical User Interface (GUI), which is implemented with the Microsoft Foundation Class Library, provides four different functions. First it builds the 0.xml file which contains user defined data. Then, the second function deals with FLAME framework, which uses the initial xml file mentioned above, to run requested simulations and generate various output results into xml files. After running FLAME, the GUI executes the xml-to-bin-converter – the third function. Thus, xml files are converted to one single binary data file. This conversion helps improving the performance of the three-dimensional graphics process. The visualisation step, the last function, written in C++, using OpenGL library, displays the incoming data and outcomes of the simulation [15].

Figure 3 shows screen shots of the GUI. Firstly when the system starts, the top window appears. Through this window it is specified the species used in the simulation and visualisation (currently Vibrio fischeri), and, optionally setting various initial values, by pressing the “Enter Data” button. The “render scene” button may be used to render output data. After introducing the initial values requested by the current experiments a simulation phase is performed by hitting the “run” button. If a graphical output is requested then going back to the top window a new graphical representation may be obtained for the last simulation.

4 EXPERIMENTS & RESULTS

The agent model presented relies on a set of chemical reactions given by Figure 1.

k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11
0.002	0.002	0.009	0.001	0.01	0.002	0.25	0.2	0.01	0.03	0.02

Table 1. Used reaction rates

According to this model each reaction specifies the reactants and products as well as a kinetic rate. The values of these constants used in the following experiments are provided in Table 1.

According to the model presentation, given in Section 2, each bacterium produces signalling molecules and proteins at a basal rate, given by reactions r1 and r2. In certain conditions, when the concentration of the signalling molecule is high enough, in certain parts of the environment, the bacteria in that area will become quorated and instead of using the basal production rate for the above mentioned chemicals will switch to another state where reactions involving the use of the complex LuxR-signal, r7 and r8, are used. In a number of experiments below we will show how QS will emerge from local interactions when environment conditions are appropriate. To simulate the process of QS occurring within a bacterium colony, the right concentration of various chemicals is necessary to be identified. The concentration threshold is given by the amount of signalling molecule.

Firstly we started with five bacteria, spread randomly in a relatively big environment such as to avoid a high cell density population in some parts of the space. The initial values of the molecules are all zero except for LuxBox, which has the value one. This experiment is repeated five times and each time, 20000 iterations were performed.

	Max signal	Max protein	r1	r2	r7	r8
RUN 1	4	4	5318	4651	0	0
RUN 2	4	6	5301	4698	0	0
RUN 3	4	4	5349	4660	0	0
RUN 4	3	4	5340	4613	0	0
RUN 5	4	4	5371	4608	0	0

Table 2. Five isolated bacteria for 20000 steps

From Table 2 it follows that only rules r1 and r2 are used over 20000 iterations in each of the five experiments and none of the r7 or r8 is ever used. This shows all bacteria keep running the non-quorated state, producing signalling molecules and proteins at a basal rate. The maximum number of signalling molecule is 4 and of the protein is 6, across all experiments and iterations. These experiments clearly show that when the cell density is lower nothing happens even over a long period of time. This will be also reinforced by the latter experiments reported in Table 3. More than this, from Figure 4, it turns out that most of the time these values are between 2 and 3 and very rarely they reach values above 3.

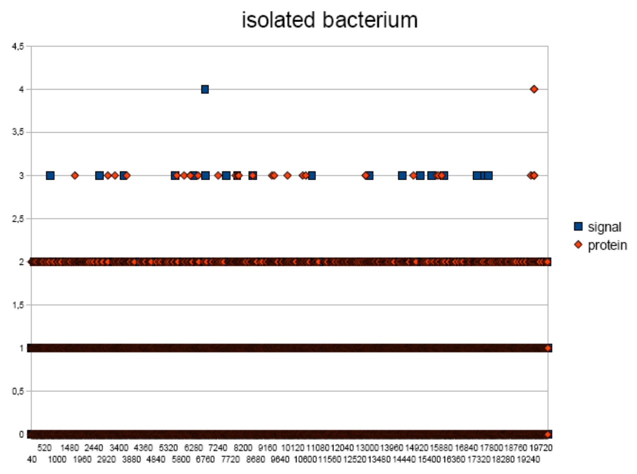


Figure 4. Signalling molecules and proteins in one isolated bacterium

From these experiments we have learned that if we consider 5 as a threshold for the number of signalling molecules that might trigger the QS, i.e., allowing rules r7 and r8 to be used, then this is coherent with individual behaviour of bacteria acting in isolation.

The second type of experiment we ran is with a colony with some special distribution of bacteria in the environment and different concentrations of the signalling molecule in the environment. All these experiments are run with 9 bacteria and 5000 steps. Five of these nine bacteria are very close to each other in one corner of the environment, whereas the other four are far away from that area. So we can compare the behaviour of the whole colony in these conditions and witness the occurrence of the emergent behaviour with respect to cell-density. In this colony we put into the bacterium that is in the middle of the group of 5 cells different initial values for the signalling molecule, so that this one, from the beginning, will be activated and will produce enough signalling molecules into the environment such as to saturate a certain part of it and to force the neighbour bacteria to accept the surplus of signalling chemicals. These initial values of the signalling molecule and protein are listed in the first column of Table 3 as a pair (x,y); x represents the number of signalling molecules and y the number of proteins. Three different experiments are reported – see Table 3. The other columns in Table 3 refer to data collected from the four bacteria surrounding the activated one and show when they become activated, the highest concentration of the signalling molecule and the number of times rules r1, r2, r7 and r8 are used. The other four bacteria are never activated, they keep producing the usual chemicals at a basal rate.

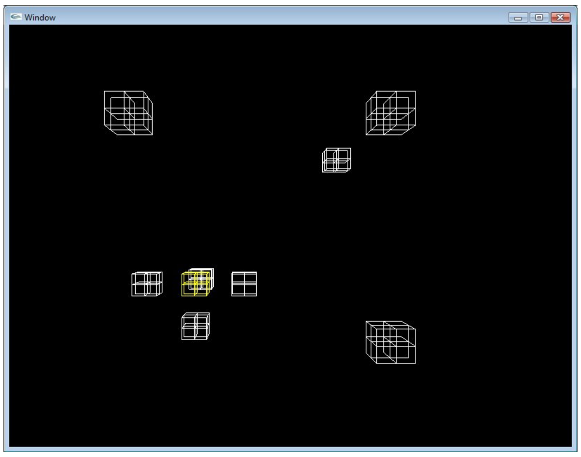
initial concentration	activation after step range	highest concentration of signalling molecules	r1	r2	r7	r8
(500, 20)	2235 - 3855	63 - 88	593 - 989	500 - 915	70 - 156	66 - 123
(1000, 20)	734 - 966	58 - 95	195 - 309	179 - 281	194 - 241	168 - 196
(5000, 100)	680 - 888	73 - 112	182 - 261	146 - 249	203 - 238	159 - 185

Table 3. Behaviour of a bacterium colony

From Table 3 one can get that the activation step of the QS mechanism is between 680 and 3855, depending on various chemical concentrations of the above mentioned chemicals. The use of rules r1, r2 on the one hand and r7, r8, on the other hand refers to the two behaviour stages of the bacteria involved, *non-activation* – only the first rules are used and *activation* – only the last two are employed.

To illustrate the evolution in time of the colony and the transformations occurring within the environment, in Figure 5, a three dimensional visualisation framework is provided. In order to keep the visualisation process as simple as possible, the agents are shown as cubes depicted with different colours. These cubes are not solid blocks, but only frames. Therewith it is easy to view all the agents in the background. The colour of a bacterium-cube is white for an inactivated bacterium and yellow for an activated one. The environment-agents are grey at the beginning of the simulation and when signalling molecules start to appear they change to a dark green colour. With this colour-change it is possible for the user to see the diffusion of the signalling molecules and how do they accumulate in some parts of the environment and then the gradual activation of various bacteria.

In Figure 5 few visualisation screenshots are represented. The first picture shows the initialised state of the simulation, when only the central bacterium in the left-bottom group is active and thereby yellow. The other bacteria are all white and they only produce signalling molecules and proteins at a low rate. Remember the active bacterium has quite a massive surplus of signalling molecule which is released into the surrounding environment and after a while will enter the bacteria nearby. The second picture shows a snapshot of the time when the environment around the central bacterium has plenty signalling molecules and one of the other four bacteria from that part becomes active, turns to yellow. The green parts around the bacteria are the environment agents, which contain signalling molecules. The brighter these cubes are, the more signalling molecules are in these places.



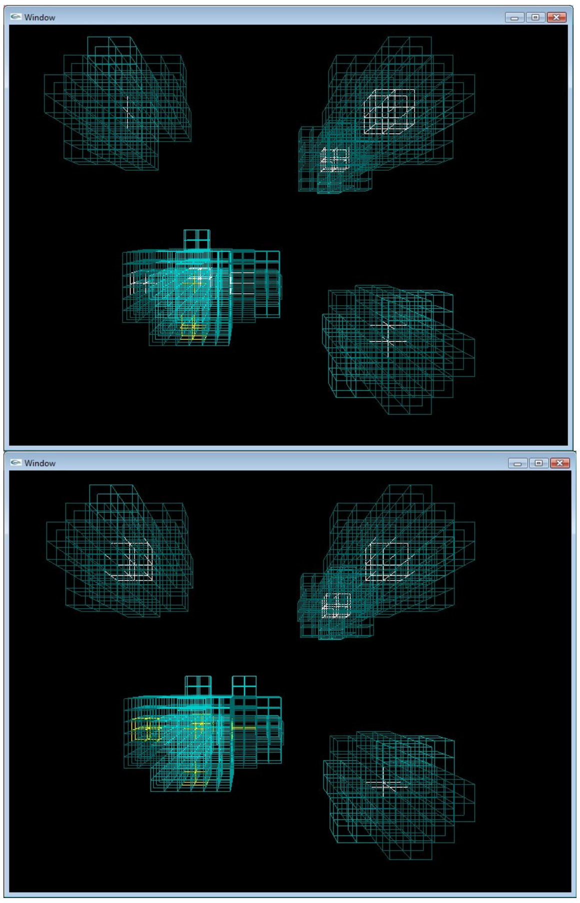


Figure 5. Screenshots of an example experiment with 9 bacteria

In the last picture all the bottom-left bacteria are activated whereas the other four remain inactivated and they will still be in that state even after 20000 steps.

These experiments clearly show a high dependency of the QS behaviour on the cell-density as well as a high stability with respect to parameters considered in these experiments.

The work reported in this paper may be also considered as a continuation of the investigation reported in [4] where QS phenomena are presented with respect to environment changes. This study represents a step forward by considering a broader range of experiments, with a well-established set of internal rules and fine-tuned parameters, within a stable range of values. The experiments clearly show when emergent behaviour occurs and how various rules and chemical concentrations contribute to this process.

5 CONCLUSIONS

The paper presents an agent based approach to simulate the behaviour of a bacterium colony with the aim of revealing some emergent properties with respect to QS phenomenon. A stochastic model based on Gillespie algorithm has been implemented for the agents running within a flexible agent platform, called FLAME. A graphical interface allowing to adjust various parameters of the model and to visualise the behaviour of the system is presented.

The current case study shows the potential of the approach to reveal emergent behaviour of a set of agents as a consequence of cell-density concentration in the environment and various ways of producing specific signalling molecules.

The model and the software platform described in this paper may be used to describe more complex systems and to identify specific behaviour patterns. This is one of our future plans and in order to make it achievable an intermediary step will be to investigate mechanisms to speed up the execution time of the simulator by migrating it on more efficient platforms, like clusters or graphical cards.

REFERENCES

[1] W. C. Fuqua, S. C. Winans, and E. P. Greenberg. Quorum Sensing in Bacteria: the LuxR-LuxI Family of Cell Density-Responsive Transcriptional Regulators. In: *Journal of Bacteriology* 176: 269-275 (1994)

[2] K. H. Nealson, T. Platt, and J. W. Hastings. Cellular Control of the Synthesis and Activity of the Bacterial Luminescent System. In: *Journal of Bacteriology* 104: 313-322 (1970)

[3] M. B. Miller and B. L. Bassler. Quorum Sensing in Bacteria. In: *Annu. Rev. Microbiol*. 55:165-199 (2001)

[4] N.Krasnogor, M. Gheorghe, G. Terrazas, S. Diggle, P. Williams, and M. Camara. An Appealing Computational Mechanism Drawn from Bacterial Quorum Sensing. In: *Bulletin of the European Association for Theoretical Computer Science*. (2005)

[5] C. M. Macal and M. J. North. Introduction to Agent-based Modeling and Simulation, In: *Argonne National laboratory, MCS LANS Informal Seminar.* (2006) Available at `http://www-unix.mcs.anl.gov/~leyffer/listn/slides-06/MacalNorth.pdf` (access 07th December 2007)

[6] C. W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. In: *Computer Graphics*, 21: (4) 25-34. (1987)

[7] U. Wilensky. NetLogo. (1999) `http://ccl.northwestern.edu/netlogo/`. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston. IL

[8] The Swarm Development Group Wiki. `http://www.swarm.org/wiki/Main_Page` (access 7th December 2007)

[9] S. Luke, C. Cioffi-Revilla, L. Panait, and K. Sullivian. MASON: A New Multi-Agent Simulation Toolkit. In: *Proceedings of the 2004 SwarmFest Workshop*. (2005)

[10] Mike Holcombe, Simon Coakley and Rod Smallwood. A General Framework for Agent-based Modelling of Complex Systems. In: *EURACE Working paper WP1.1* (2006)

[11] F. Bernardini, F.J. Romero-Campero, M. Gheorghe, and M.J. Pérez-Jiménez. A Modeling Approach Based on P Systems with Bounded Parallelism. In: *Lecture Notes in Computer Science*. Springer. 4261/2006. Berlin/ Heidelberg. 49-65. (2007)

[12] D. T. Gillespie. Exact Stochastic Simulation of Coupled Chemical Reactions. In: *The Journal of Physical Chemistry* 81: (25) 2340-2361. (1977)

[13] `http://www.flame.co.uk` (access 27th November 2007)

[14] `https://trac.flame.ac.uk/wiki/` (access 29th November 2007)

[15] M. Woo et al. OpenGL Programming Guide. Third Edition., published by Addison-Wesley Longman, Amsterdam. Page 2 f. (2000)

Trained Particle Swarm Optimization for Ad-Hoc Collaborative Computing Networks

Shahin Gheitanchi, Falah Ali, Elias Stipidis

Abstract. Distributed processing is an essential part of collaborative computing techniques over ad-hoc networks. In this paper, a generalized particle swarm optimization (PSO) model for communication networks is introduced. A modified version of PSO, called trained PSO (TPSO), consisting of distributed particles that are adapted to reduce traffic and computational overhead of the optimization process is proposed. The TPSO technique is used to find the node with the highest processing load in an ad-hoc collaborative computing system. The simulation results show that the TPSO algorithm significantly reduces the traffic overhead, computation complexity and convergence time of particles, in comparison to the PSO.

1 INTRODUCTION

Deployment of wireless communication services based on collaborative computing has recently been considered by researchers in different areas [1-3]. Collaborative computing requires ad-hoc networking and efficient distributed processing techniques to perform complex tasks in a reasonable time.

Particle Swarm Optimization (PSO) [4] is a well known pervasive optimization technique in artificial intelligence that is based on behavior of social animals such as bees and birds. In the PSO technique each individual member of social colony, like a bird, is called a particle. For example, we observe PSO in a swarm of birds in a field. Their goal is to find the location with the highest density of food. Without any prior knowledge of the field, the birds begin their search in random locations with random velocities. Each bird can remember the locations that it has found more food, and somehow knows the locations where the other birds found large amount of food. Each bird has the choice to decide between returning to the location where it had found the most food itself, or exploring the location reported by others to have the most food, the bird accelerates in both directions somewhere between the two points depending on whether nostalgia or social influence dominates its decision. Along the way, a bird might find a place with more food than it had found previously. It would then head to this new location as well as the location of the most food found by the whole swarm. Occasionally, one bird may fly over a place with more food than have been found by any other bird in the swarm. The whole swarm would then head toward that location in additional to their discovery. Soon, all or most of the birds gather around the location where the highest density of food is there.

To increase the efficiently and performance of different OSI layers [5], the PSO technique has been used in the literature for various scenarios [6–10]. Most of PSO applications in communications have been focused on clustering in ad-hoc networks aiming to minimize energy consumption [6-8]. In [6] the authors have applied PSO to cluster head selection and in [7] it has been used for distance based clustering of wireless sensor networks. Also in [8] the algorithm was used to optimize the cost and coverage of clustering in mobile ad-hoc networks. Many other applications for PSO in communications such as IP multicasting and channel assignment have been mentioned in [9]. Utilizing the PSO in ad-hoc networks increases flexibility, adaptation and robustness. While being simple, it can also introduce enormous traffic and computation overhead to the network and may lead to long convergence time. The traffic overhead is the number of extra packets needed to be transmitted over the network to accomplish the optimization task. The computation complexity overhead is the time (number of iterations) needed by a node to process the particles gathered over it.

In this paper, we introduce a generalized PSO model for communication networks. Then, based on the PSO model, we propose trained PSO (TPSO) for ad-hoc networks as a new technique to support collaborative computing networks. Using the proposed models, we simulate PSO and TPSO techniques in an ad-hoc network to find the node with the highest processing load. Finally, we compare the traffic overhead, computation complexity overhead and convergence time of the techniques.

2 GENERALIZED PSO MODEL

PSO system model consists of *P* number of particles and unique particle IDs (PIDs) which are randomly distributed over the problem solution space. The solution space, *S*, is the set of all possible solutions for the optimization problem. Depending on the problem, the solution space can have *N* number of dimensions, S^N, where each dimension contains different number of elements. Each particle is capable of measuring the suitability of solutions by using the fitness function $f(s^1, s^2, ..., s^n)$, where $0 < n \leq N$ and $s^n \in S^N$. All particles use unique and identical fitness function to be able to assess the suitability of a solution. The objective of the optimization is to find a set of $\hat{S} \subset S$ to maximize the fitness function

$$\hat{S} = \text{Argmax}\, f(s^1, s^2, ..., s^n) \qquad (1)$$

Each particle stores the value and location of the best solution found so far, called the local best (LB). Also all particles are aware of the value and location of the best solution found by all other particles, called the global best (GB). Assuming synchronized timing and unique speed among the particles, the optimization is performed during Γ iterations. At each iteration the particles compare the LB and the GB to choose a direction independently based on the distance differences from current location to the GB and to the LB locations. The distance between two locations, (s_1^1, s_2^1) and (s_1^2, s_2^2), for $N = 2$ is given by

$$d = \sqrt{(s_1^2 - s_1^1)^2 + (s_2^2 - s_2^1)^2} \tag{2}$$

Particles consider nostalgia, w_n, and social influence, w_s, for deciding their directions. The weights, w_n and w_s, describe the importance of nostalgia and social influence for particles, where $w_n + w_s = 1$. We define the following expression for deciding the direction

$$\text{direction is} \begin{cases} \text{LB} & \text{if} \quad (w_n d_{LB} - w_s d_{GB}) \le 0 \\ \text{GB} & \text{if} \quad (w_n d_{LB} - w_s d_{GB}) > 0 \end{cases} \tag{3}$$

Where, for each particle, d_{LB} is the distance from the current location to the LB and d_{GB} is the distance from the current location to the GB. After specifying the direction the particle moves toward the decided destination which is the location of the LB or the GB. During the optimization process, the GB is updated when a solution with higher fitness value is found by a particle. After Γ iterations the particles gather (or converge) on the location with the highest fitness value and the algorithm terminates which is referred to as the termination criteria 1. When the particles converge, the value of the GB is considered as the solution of the optimization problem. To avoid infinite loop in cases of having more than one GB value and also managing the execution time of the PSO algorithm, we set a relatively large number as the maximum iteration number, $\Gamma_{\max}$ in compare with elements of solution space. When the process is stopped by reaching $\Gamma_{\max}$, called termination criteria 2, the GB with highest population of particles over it is chosen as the solution for the optimization problem. Table 1 shows the general PSO algorithm.

Table 1. The generalized PSO algorithm

```
1: Initialize and distribute particles
2: Loop while not (termination criteria 1 and 2)
3:     For each particle:
4:         If (LB > GB)
5:             Replace GB and LB
6:         Calculate LB
7:         Decide the direction towards LB or GB
8:         Move to new position
9:     End of for
10: End of loop
```

3 TPSO FOR AD-HOC COLLABORATIVE COMPUTING

Because of distributed nature of particles, the proposed PSO model is suitable for efficient distributed processing with different objectives. Figure 1 shows the distributed nodes of an ad-hoc collaborative computing system. The movement of particles in an ad-hoc network introduces high traffic and computation complexity overhead. Also it may take a long time to converge. The traffic overhead is caused by movement of particles and their related information such as history of the LB. To reduce the overheads, we introduce the TPSO technique. TPSO is an improved PSO algorithm which the values of w_n, w_s and P are defined based on the requirements of the system using a training system. The idea is to adapt the particles behavior in order to benefit from the system limitations such as limited number of routes that a particle can take or the number of particles that can be over a single solution (node). We assume a perfect training system is responsible for training the particles of the PSO algorithm. As it will be shown in the simulation results, training the particles reduces the traffic and computational complexity overheads and also significant reduces the convergence time.

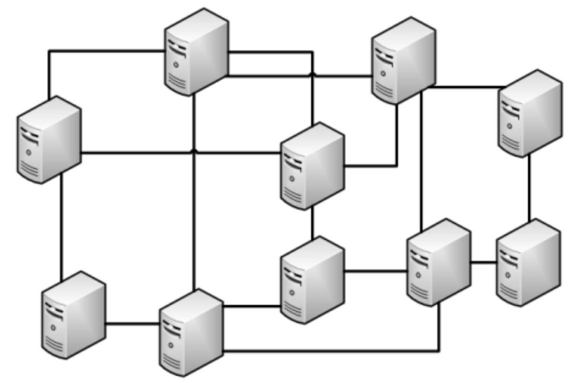


Figure 1: Ad-hoc collaborative computing network model.

In the collaborative computing system we consider an ad-hoc network consisting of randomly distributed nodes in a two-dimension ($N = 2$), X-Y, landscape. The nodes in the network support multi-connections to their neighboring nodes and are able to transmit the data to the destination using shortest path routing method. The solution space is equal to positions of nodes and is stored in a sparse matrix as following

$$S_{x,y} = \begin{cases} 1 & \text{if a node exist in } x \in X,\ y \in Y \\ 0 & \text{else} \end{cases} \tag{4}$$

The distance is measured using eq. 2 and the number of particles is less than the number of nodes. As it will be shown in the simulation results, the number of particles in PSO contributes to the convergence time of PSO algorithm but in TPSO the convergence time does not significantly change by the number of particles. However, low number of particles in the system may lead to sub-optimal results.

At the beginning of the TPSO process, the particles are randomly distributed among the nodes. In the network, the packets move only through the single available route between each two neighboring nodes. Since the solution space is equal to the position of nodes, and is a sparse matrix, it is not expected to find any solution between two neighboring nodes. Therefore, the movement of particles between two neighboring nodes that is caused by uncertainty between nostalgia and social influence will not lead to finding of a new LB or GB value. By manual training, we force the particles to always follow the social influence (choosing the GB as the next destination) using the following configuration: $w_s = 1$, $w_n = 0$. This configuration will avoid redundant movements of the particles between two neighboring nodes; hence it will reduce the traffic and computation complexity overhead. Furthermore, as it will be shown in the simulation results, the particles converge in a constant number of iterations. This is because of constant maximum distance between every two nodes, movement of all particles towards an identical node and hopping from one node to another in a single iteration.

Figure 2 shows the flowchart of the TPSO algorithm for an ad-hoc collaborative computing network. The particles are implemented on each node using an identical software agent, called the particle process (PP). It is responsible to calculate, compare and update the LB and the GB values as well as moving the particle towards GB. The GB updates are done using a broadcast algorithm in the network layer. Since the updating is performed occasionally, we neglect the caused overheads. The PP of a node runs only when at least one particle is over that node. Therefore, increasing number of particles over a node will increase the computational complexity overhead. Particles move between two nodes by using a flag, carried by the data packets circulating in the network. The flag indicates when to run a PP process in a node and also is used for counting the PIDs over a node. Since particles move among the nodes using the data packets, their movement and directions depend on the availability of connection among the nodes.

In TPSO, all particles on a node have similar destination which is the GB location or the next hop towards the GB location. To further reduce the traffic over head and computation complexity on a node, the particles are batched in single super particle which is the aggregation of all the particles on the node but with a new PID that is known to the PP processes. Then the super particle acts similar to the normal particle in the network and at each node it is treated as sum of particles when calculating the number of PIDs in PP. Using super particles will gradually reduce the number of particles in the system, *P*, as the TPSO process continues. The TPSO terminates when one of the termination criteria, explained in section 2, is met. Reducing the network traffic will reduce the computation overhead on each node by requiring fewer packets to be processed. Using the big-oh notation, we assume the computation complexity of the PSO on a single node is in order of $O(g(\Gamma))$ where $g(\Gamma)$ is the complexity function for Γ iterations on each node. The complexity will increase to $O(Qg(\Gamma))$ when Q number of particles ($Q < P$) overlap on the node. In TPSO, since Q and Γ are reduced by using the super particles, the PP will run fewer times and computational complexity over a node will decrease.

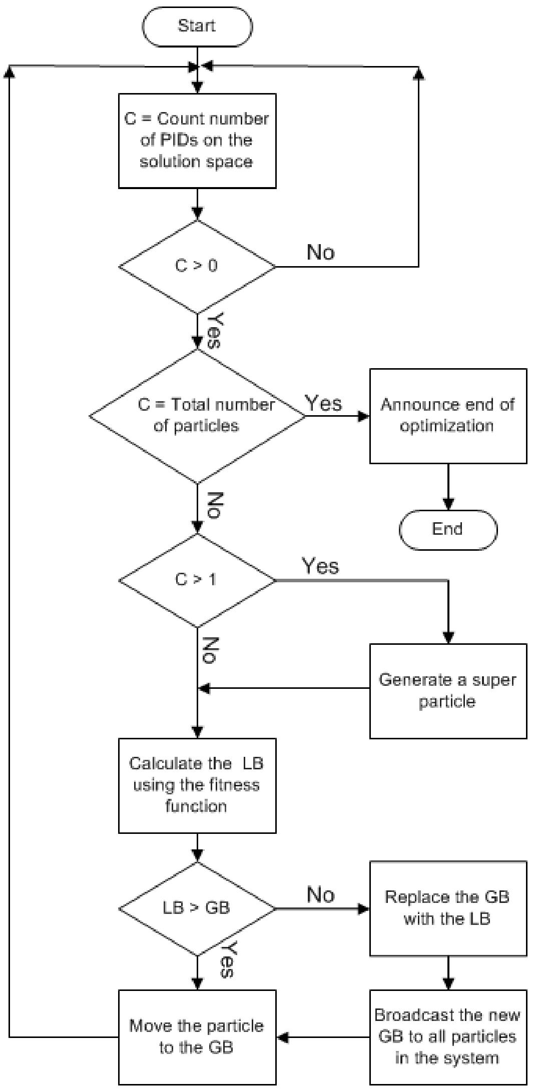

Start
C = Count number of PIDs on the solution space
C > 0
No
Yes
C = Total number of particles
Yes
Announce end of optimization
End
No
C > 1
Yes
No
Generate a super particle
Calculate the LB using the fitness function
LB > GB
No
Replace the GB with the LB
Yes
Move the particle to the GB
Broadcast the new GB to all particles in the system

Figure 2: Flowchart of the TPSO algorithm for ad-hoc collaborative computing network.

4 SIMULATION RESLUTS

In this section we consider an ad-hoc collaborative computing network with 50 nodes in a 100 by 100 landscape and 30 particles which are randomly distributed over the nodes. The nodes use shortest path routing algorithms for transmitting packets between the nodes. We simulate the proposed PSO and TPSO algorithms to find the node with the highest processing load. The results can be fed to different algorithms such as loading-based routing algorithm

[11]. The fitness function, *f(x,y),* is equal to the load of a node in location of *(x,y)*. The load of a node is a measure of number of tasks (i.e. packets) that needs to be processed and we assume it remains constant during the optimization process. The processing load of a node in *(x,y)* with *U* number of task queues, is given by

$$f(x,y)=\sum_{u=1}^{U} L_{x,y,u} \qquad (5)$$

where $L_{x,y,u}$ is the size of the *u*-th task queue. In Table 2 we introduce the packets used in the system Assuming that each data occupies only one byte, the size of packets for each packet type is calculated. We do not consider the overhead caused by the routing protocol of network layer.

Table 2.
PSO and TPSO packet sizes and description used in the simulation.

Packet type	Packet size in TPSO	Packet size in PSO	Description
GB_Broadcast	2	2	For broadcasting the value and location of the GB.
P_Move	1	4	For moving the particle from one node to another node. For PSO it contains PID, LB location and value and destination address. For the TPSO it only contains PID.
Terminate	1	1	A unique packet to indicate optimization termination

Figure 3 shows a snapshot of the proposed TPSO algorithm. The weights on each node represent the measure of processing load on that node and the distributed circles on the nodes show the particles. As the process progresses, the particles converge over the node with the highest load. Based on the termination criteria explained before, the algorithm broadcasts the found solution to the other nodes when all particles have converged over a node or maximum number of iterations has been reached.

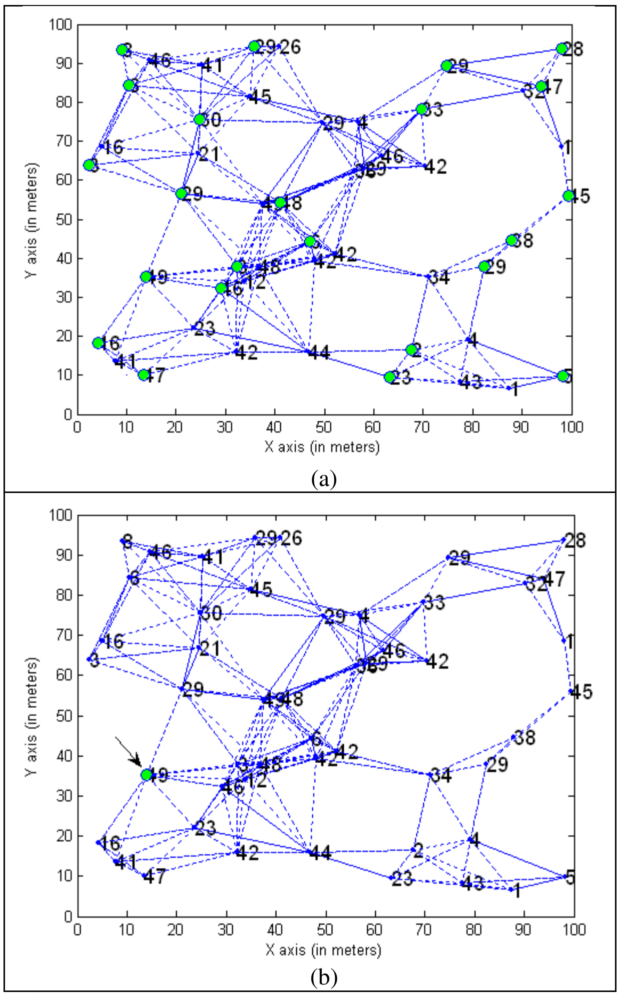


Figure 3: Snap shot of TPSO in ad-hoc collaborative computing network with 50 nodes and 30 particles showing particles a) before convergence, b) after convergence.

Figure 4 shows the difference of the average traffic overhead caused by P_Move packets of PSO and TPSO in the mentioned ad-hoc network. The reduction of the traffic overhead is due to carriage of less data from node to node. As a result of training all the particles to always move towards the node with the GB value and do not return to LB location. However the GB location may change during the optimization process but at each interval all the particles in the system are heading towards the same destination which the current GB. As mentioned before, in TPSO, when there is more than one particle over a node, they are considered as one super particle. Since each super particle is treated similar to a particle, using super particles will reduce number of packets needed to be transmitted.

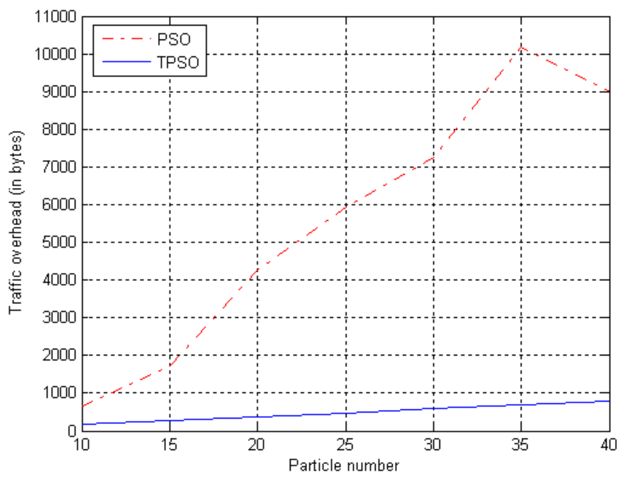


Figure 4: Comparison of PSO and TPSO traffic overheads for different number of particles over 50 distributed nodes.

In figure 5 we compare the average convergence time for PSO and TPSO based on our simulation results for different number of particles. As explained in section 3, when using TPSO the particles converge over the optimization solution with near constant number of iteration in comparison to PSO.

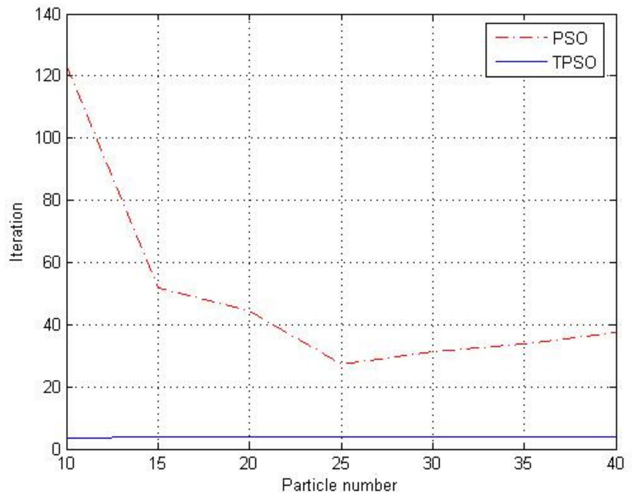


Figure 5: Comparison of convergence speed for TPSO and PSO techniques for different number of particles over 50 distributed nodes.

5 CONCLUSION

In this paper we introduced a generalized PSO algorithm for communication networks. Using the introduced PSO model, we proposed TPSO as an efficient optimization algorithm for ad-hoc collaborative computing networks. The PSO and TPSO techniques were simulated over distributed nodes to find the node with the highest processing load. The simulation results show the convergence time of TPSO is almost constant while the traffic and computational complexity over a node is reduced in comparison to PSO.

REFERENCES

[1] S. Konomi, S.Inoue, T. Kobayashi, M. Tsuchida, M. Kitsuregawa, "Supporting Colocated Interactions using RFID and Social Network Display," IEEE Pervasive Computing, Jul.- Sep.2006.

[2] P. K. McKinley, C. Tang and A. P. Mani, "A Study of Adaptive Forward Error Correction for Wireless Collaborative Computing," IEEE Transaction on parallel and distributed systems, Vol. 13, No. 9, Sep. 2002.

[3] C. Lai, W. Polak, "A Collaborative Approach to Stochastic Load Balancing with Networked Queues of Autonomous Service Clusters," MobCops Workshop of the 2nd IEEE Collaborative Computing Conference (CollaborateCom 2006), Atlanta, GA, USA, Nov. 2006.

[4] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in Proc.IEEE Conf. Neural Networks IV, Piscataway, NJ, 1995.

[5] A. S. Tanenbaum, "Computer Networks," Prentice Hall Ptr; 4 ed, 2002.

[6] Tillett, J. Rao, R. Sahin, F. , "Cluster-head identification in ad hoc sensor networks using particle swarm optimization," IEEE int. conf. on Personal Wireless Communications Proc., 2002,pp. 201- 205.

[7] S.M. Guru, S.K. Halgamuge, S. Fernando, "Particle Swarm Optimisers for Cluster formation in Wireless Sensor Networks," International conference on intelligent sensors, sensor networks and information processing proc., 2005, pp. 219- 224.

[8] X. Wu, S. Lei, J. Yang, X. Hui, J. Cho, S. Lee, "Swarm Based Sensor Deployment Optimization in Ad Hoc Sensor Networks," Int. conf. on embedded software and systems (ICESS), 2005.

[9] W. Ping, W. Guang-Zing, Z. Yang-Yang, "Particle Swarm Optimization Approach of Solving Communication Optimization Problems," Journal of Northeastern University, Natural Science (China). Vol. 25, no. 10, Oct. 2004, pp. 934-937.

[10] S. Gheitanchi, F. H. Ali, E. Stipidis, "Particle Swarm Optimization for Resource Allocation in OFDMA," IEEE DSP07 conference proceeding, Jul. 2007, pp. 383-386.

[11] H. Hassanein, A. Zhou, "Routing With Load Balancing in Wireless Ad hoc Networks", Proc. of 4th ACM international workshop on modeling, analysis and simulation of wireless and mobile systems, 2001.

Controller-Agent based approach for Solving Distributed Constraint Problem

Sami AL-MAQTARI[1] **and Habib ABDULRAB** [1]

Abstract. The interesting domain of constraint programming has been studied for years. Using Constraint Satisfaction Problem in association with Multi-Agent System has emerged the research in a new field known as Distributed Constraint Satisfaction Problem (DCSP). Many algorithms are proposed to solve DCSP. Inspired from ABT algorithm, we introduce in this paper our algorithm for solving DCSP where we divide agents in the system into two groups: Variables' and Controller Agents, which allow reformulating of different inter-agent communication algorithm in our framework. This division allows the separation between the constraints verification and other functions of agents. The proposed algorithm is not only capable of treating binary constraints; it can be used easily in order to treat non-binary constraints. This facility gives also the possibility of grouping constraints in order to form a set of quasi-independent sub-problems. These sub-problems may be inter-connected by some common variables. The instantiation of these variables can be done by negotiation in order to separate the sub-problems into totally independent ones.

1 INTRODUCTION

Constraint Satisfaction Problem or CSP is a very interesting paradigm for modeling in real life. A lot of real world problems can be described and modeled as a set of variables where their values are restricted by a set of constraints. Many methods and techniques are already in use in order to treat this kind of problems. Backtracking and arc-consistency and their alternatives are already discussed over and over in literature [1, 2].

Mixing CSP with the autonomy of agents in a Multi-Agent System emerges the birth of a sub-branch of CSP research known as Distributed CSP or DCSP. Multiple methods are used to solve a DCSP. Among them we find the Asynchronous BackTracking (ABT) proposed by Yokoo and its alternative [3-7] that we inspired our approach from.

This Approach is very adequate for modeling a specific kind of problems like water usage in agricultural domain. In such system constraints are suitable for modeling the ecosystem equations. However, they cannot be used in order to model the autonomous behavior of the different actors in the system (like farmers and resource owner, etc.); where multi-agent systems are more convenient.

We propose a new framework based on two kinds of agents: Variables' agents and Controller Agents. By defining these two kinds of agents we separate between the treatment of constraints and the other functionalities of the agents in the system. Also, it allows dividing a general constraint problem into multiple sub-problems easier to be treated. This algorithm is firstly introduced in our paper [8] for a proposed model intended to be used for water agricultural usage management to be applied to the region of Sadah in Yemen.

In the next sections we start by giving some basic definitions then, we introduce our proposition for an alternative algorithm for solving DCSPs.

2 Definitions

We start by giving some definitions of CSP and DCSP:

2.1 Constraint Satisfaction Problem (CSP)

Formally, a Constraint Satisfaction Problem (CSP) is a triple (V, D, C) where:

- $V = \{v_1, \ldots, v_n\}$ is a set of n variables,
- a corresponding set $D = \{D(v_1), \ldots, D(v_n)\}$ of n domains from which each variable can take its values from,
- and $C = \{c_1, \ldots, c_m\}$ is a set of m constraints over the values of the variables in V. Each constraint $c_i = C(V_i)$ is a logical predicate over subset of variables $V_i \subseteq V$ with an arbitrary arity $k : c_i(v_a, \ldots, v_k)$ that maps the cartesian product $D(v_a) \times \ldots \times D(v_k)$ to $\{0,1\}$. As usually the value 1 means that the value combination for $v_a, \ldots, v_k$ is allowed, and 0 otherwise.

A solution for a CSP is an assignment of values for each variable in V such that all the constraints in C are satisfied.

Constraints involving only two variables are called binary constraints [9]. A binary constraint between x_i and x_j can be denoted as c_{ij}.

Although most of real world problems are represented by non-binary constraints, most of them can be binarized using techniques such that dual graph method and hidden variable method [10]. Translating non-binary constraints into binary ones allows processing the CSP using efficient techniques adapted only for binary constraints. However, this translation normally implies an increase in number of constraints.

[1] LITIS Lab. – INSA of Rouen – France. Email: {sami.almaqtari, abdulrab}@ insa-rouen.fr.

2.2 Distributed Constraint Satisfaction Problem (DCSP)

A Distributed Constraint Satisfaction Problem (DCSP) is a CSP where the variables are distributed among Agents in a Multi-Agent System (MAS). A DCSP can be formalized as a combination of (V, D, C, A, λ) where:

- V, D, C are the same as explained for a normal CSP,
- $A = \{a_1, \dots, a_p\}$ is a set of p agents,
- and $\lambda : V \to A$ is a function used to map each variable v_j to its owner agent a_i.

Each variable belongs to only one agent, i.e. $\forall v_1,\dots,v_k \in V_i \Leftrightarrow \lambda(v_1) = \dots = \lambda(v_k)$ where $V_i \subset V$ represents the subset of variables that belong to agent a_i. These subsets are distinct, i.e. $V_1 \cap \dots \cap V_p = \phi$ and the union of all subsets represents the set of all variables, i.e. $V_1 \cup \dots \cup V_p = V$.

The distribution of variables among agents divides the constraints set C into two subsets according to the variables involved within the constraint. The first set is the intra-agent constraints C_{intra} that represent the constraints over the variables owned by the same agent $C_{intra} = \{C(V_i) \mid \lambda(v_1) = \dots = \lambda(v_k), v_1, \dots, v_k \in V_i\}$.

The second set is the inter-agent constraints C_{inter} that represent the constraints over the variables owned by two or more agents. Obviously, these two subsets are distinct $C_{intra} \cap C_{inter} = \phi$ and complementary $C_{intra} \cup C_{inter} = C$.

The variables involved within inter-agent constraints C_{inter} are denoted as interface variables $V_{interface}$. Assigning values to a variable in a constraint that belongs to C_{inter} has a direct effect on all the agents which have variables involved in the same constraint. The interface variables should take values before the rest of the variables in the system in order to satisfy the constraints inside C_{inter} firstly. Then, the satisfaction of internal constraints in C_{intra} becomes an internal problem that can be treated separately inside each agent independently of other agents. If the agent cannot find a solution for its intra-agent constraints, it fails and requests another value proposition for its interface variables.

To simplify things, we will assume that there are no intra-agent constraints, i.e. $C_{intra} = \phi$. Therefore, all variables in V are interface variables $V = V_{interface}$.

Many techniques are used to solve DCSPs. Mainly we mention the Asynchronous Backtracking (ABT) algorithm that was proposed by Yokoo [3, 11-13] and some of its alternatives [4, 5]. In the following section we introduce another alternative based on the concept of Controller Agent, which we propose, in order to validate and test inter-agent constraints.

2.3 Over-constrained problem and constraint relaxation

A CSP is called an over-constrained problem if no possible combination exists that satisfies all constraints. In other words, at least one constraint can not be satisfied. Solving such problem implies neglecting some constraints in the system (constraint relaxation [14]). This can be achieved by dividing constraints according to some kind of preference levels [15]. Constraint types may vary from hard to soft constraints according to their importance. The hard ones represent the constraints that should absolutely be satisfied while the soft constraints may be neglected. In this case we search the optimal solution in trying to maximize the number of satisfied soft constraints. We say that no solution exists if we cannot satisfy all the hard constraints.

3 Controller Agent approach

Here we will explain how to extend the original DCSP-based model of multi-agent cooperation. We will describe a structure of DCSP based on the idea of dividing agents into two types: Variables' Agent and Controller Agent. Originally introduced in our former works [8, 16], a Variables' Agent is an agent who possesses one or more variables from the set V and a subset of intra-agent constraints from C_{intra}, while the role of Controller Agents is to encapsulate the inter-agents constraints. For short, we will use the term VAgent as an abbreviation for Variables' Agent and CAgent for Controller Agent.

CAgents take in charge the responsibility of verifying the constraints satisfaction. In other words, from the beginning all global constraints are distributed among an initial set of CAgents $CA = \{ca_1, \dots, ca_q\}$ where $1 \le q \le m$ and m is the number of inter-agent constraints. In the centralized form of DCSP all constraints in C_{inter} are grouped inside one CAgent ($q = 1$), while in its simplest form each CAgent contains only one constraint from C_{inter} ($q = m$).

Figure 1 shows a constraint network of a set of agent related by binary constraints either directly (a) or via CAgent (b). As in a formal constraint network, Figure 1 (a) shows some constraints that represent relations directed from an agent (called proposing agent) to another (called evaluating agent) according to a predetermined order. The resolution is done by passing value propositions according to the directions of arcs from proposing agent to evaluating agent.

In our approach shown in Figure 1 (b), relations between agents are represented by links directed from a VAgents to a CAgents. These links indicate that the VAgent has at least one variable that is involved in a constraint of the CAgent.

The CAgent supervises one or more constraints. This means that we can group more than one constraint into Controller to form a kind of local problem as shown in Figure 1 (b). The propositions of values for variables are sent by VAgent to be received and tested by CAgents. VAgents are given different priority degrees (by default, their order of creation). Such application of VAgents and CAgents facilitates seamless combination of MAS and CSP.

Using the concept of VAgents and CAgents needs re-formulating inter-agent communication algorithm in our framework. In the next section we will illustrate how the modification of well-known asynchronous backtracking algorithm by Yokoo [3, 11] can be easily achieved. For simplicity reasons, some limitations are made in this paper and they are not an obligation: we will consider only the case where a VAgent has one variable only. In the same manner, we will treat the case where a CAgent contains one binary constraint only. The proposed approach can be easily extended and applied to other cases without these limitations as we will see later in this paper.

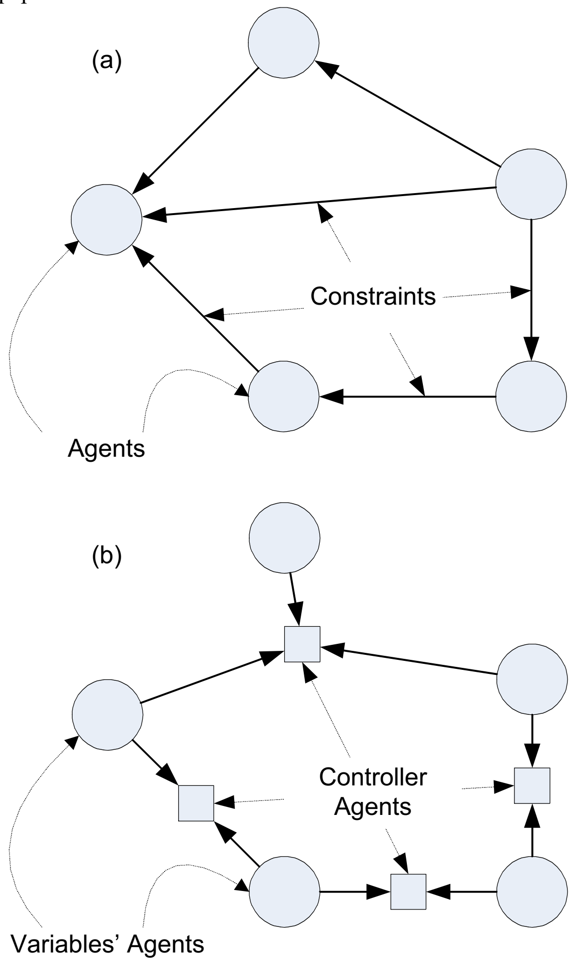


Figure 1 Constraint Network (a) without or (b) with Controller Agent

3.1 Algorithms

As mentioned before, the proposed algorithm may be considered as a modified version of the ABT algorithm of Yokoo[3, 4]. The proposed algorithms of both types of agents are divided into two stages:

- **Domain reducing stage**

This stage implies a preprocessing of variables' domains. The result is reduced domains by eliminating values that would be surly refused from them. This is done as follows:

1. A VAgent sends information concerning the domain of its variable to all linked CAgents. The message takes the form of (variable, domain).
2. After receiving the domains of all variables involved in its constraint, the CAgent use bound consistency algorithm in order to reduce these domains to new ones according to its local constraint(s). Then, the controller sends these domains back to their VAgents.
3. Every VAgent receives the new domains sent by CAgents and combines them (by the intersection of received domains) in order to construct a new version of its variable domain.
4. If any new version of a variable domain was empty then we can say that this DCSP is an over-constrained problem. In this case, the system signals a no-solution to user (failure). Another solution can be investigated by using constraints relaxation [14, 15], in which a VAgent It returns to an older version of the domain and reconstruct a new version after neglecting the domains sent by the CAgent that represents the soft constraints that the system may violate according to certain constraint hierarchy [15]. On the other hand, if all variables end with single-value domains then one solution is found. Otherwise, the domain reducing stage is repeated as long as we obtain a different new version of a variable domain. When domain reducing is no longer possible (no more change in variables' domains); we can proceed to the next stage.

- **Value proposition and validation stage**

In this stage VAgents make their propositions of values to related CAgents to be tested. Value proposing can be considered as a domain inform message in test mode. This proceeds as follows:

5. From now on, every VAgent starts instantiating values for its variable according to the new domains. It sends this proposition to the related CAgents.
6. The CAgent chooses the value received from the VAgent with the highest priorities. This value is considered as a domain with a single value. CAgent uses bound consistency algorithm as in the previous stage to reduce other variables' domains. These new domains are sent to their VAgents to propagate domains change.
7. Like in the previous stage, if all variables end with single-value domains then one solution is found. Unlikely, if the result of this propagation was an empty domain for any variable then the proposed value is rejected and another value is requested. If no more value can be proposed then system signals a no-solution situation to user.
8. If the result of the domain propagation was some new reduced domains with more than one value then steps 5-7 are repeated Recursively with the value proposed by the VAgent that have the next priority.

3.2 Advantages and disadvantages

Main advantage of the system is the possibility of treating non-binary constraints simply without need of modification. The algorithm is still working for non-binary constraints in the same manner. This allows avoiding the need for constraints translation into binary ones. Extending the algorithm to cover non-binary constraints will be explained later in this paper.

The possibility of treating non-binary constraints allows us to gather constraints to create subsets of constraints where each subset contains a group of constraints that could be related and included inside one CAgent. Searching a solution for this group represents a sub-problem of the general one. In one of its extreme case, this grouping allows centralizing all the resolution (in case where all constraints are included inside one CAgent). In normal case, this allows dividing constraints into groups that are related by some common variables. Giving value for these variables separates these groups of constraints and forms a set of sub-problems that are totally independent. Many methods [17, 18] are proposed in order to partition the problem to allows processing it in parallel manner. These methods can be used with our algorithm in order to assign each partition of the problem to a CAgent.

A main disadvantage of the proposed algorithm is the increase in agent number. In fact, using CAgent for each constraint (mostly binary ones) may overload the system because of the large total number of agents (the increase according to the number of constraints). However, the benefits of grouping constraints into subsets and putting each group inside a controller can compensate the increase in number of agents. We still have some improvement to do in order to increase the performance of the algorithm. This performance can be improved by taking in consideration some enhancements that will be discussed later.

3.3 Algorithm Correctness

For any CSP we can encounter one of three cases: if it is an over-constraint problem and then the solution does not exist. Otherwise, one solution or more exist. Here, we will explain how this approach can detect these cases.

As described in step 4 in the domain reduction stage of the algorithm, the detection of an empty domain for any variable means that this DCSP is over-constrained problem and no solution exists. In the other side, if all variables after the domain reduction stage have only one single value domains, then this signifies that only one solution exists which is the combination of all domain (single) values.

Otherwise, one or more solutions may exist. By considering value proposing and validating stage of the algorithm, we consider value proposing as domain informing with a single value in test mode. We call it test mode because unlike domain reduction stage here we can request backtracking.

To understand backtracking in our algorithm we can imagine a situation where a VAgent V1 proposes a value for its variable. This value is considered by the related CAgent C as a single-value domain and would propagate to other related VAgents. If another VAgent V2 ends with an empty domain for its variable because of V1 proposal then it would report a backtrack signal to the CAgent C. in this case, CAgent C will request another value from VAgent V1 and VAgent V2 should retrieve its previous variable domain before domain propagation.

The recursive nature of value proposing stage results always either single-value domains for all variables which means that a solution exists or an empty domain case which means that no solution exists.

3.4 Example

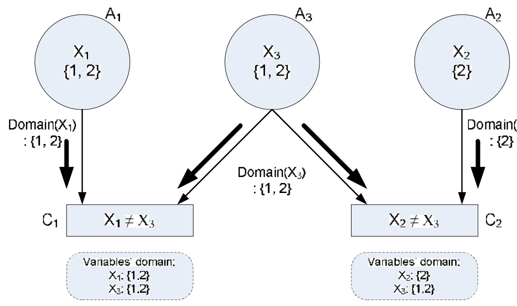


Figure 2 domain sending

The following figures show an example of the proposed algorithm. Three VAgents A_1, A_2 and A_3 having three variables X_1, X_2 and X_3 respectively. Each variable has its respective domain. These agents are related to two CAgents C_1 and C_2 which contain two constraints over the variables.

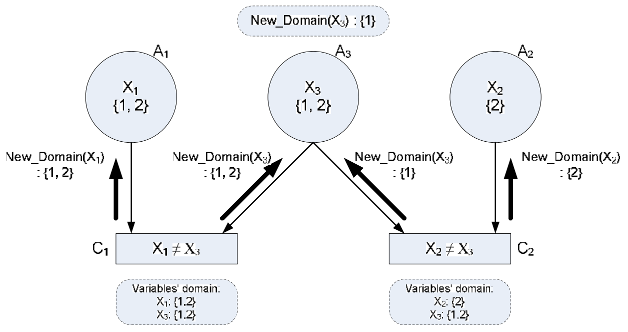


Figure 3 returning reduced domains

In Figure 2 the VAgents start sending information about the domains of their variables to the CAgents. The CAgent calculates the validity of these domains according to its constraints. In Figure 3, the controller C_1 cannot reduce the domains of X_1 and X_3. However, the controller C_2 can reduce the domain of X_3 and send back the new domin.

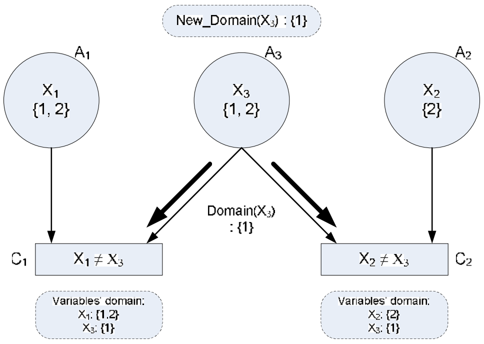


Figure 4 propagation of domain change

In order to propagate the domain change in the agent A_3, it will resend its domain to the related controllers as shown in Figure 4. This propagation also informs the controllers about the new domain of the variables.

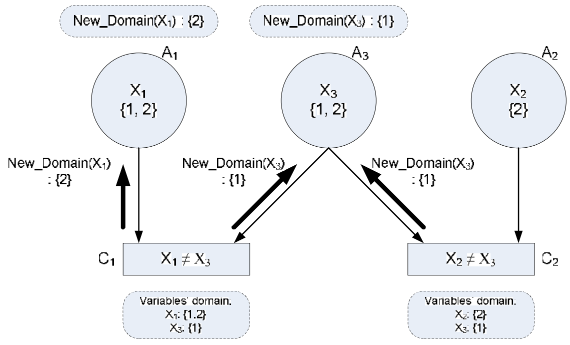


Figure 5 continue domain reduction

The domain change propagation can cause another domain change in another VAgent. In Figure 5, agent A_1 receives another proposition for its variable new domain. While this propagation has no effect on the domain of the variable in agent A_2.

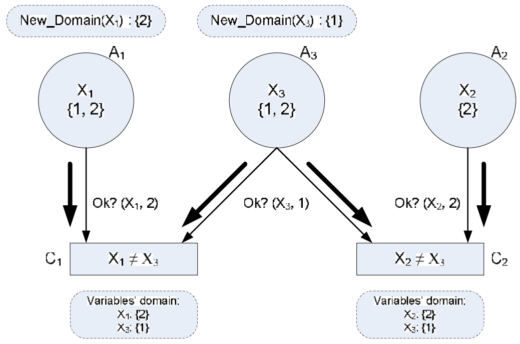


Figure 6 value proposing

Each VAgent receives the reduced domain and creates a new domain by the intersection between the returned domains from each controller. In Figure 3, Agent A_3 has created the new domain $\{1\} = \{1,2\} \cap \{1\}$.

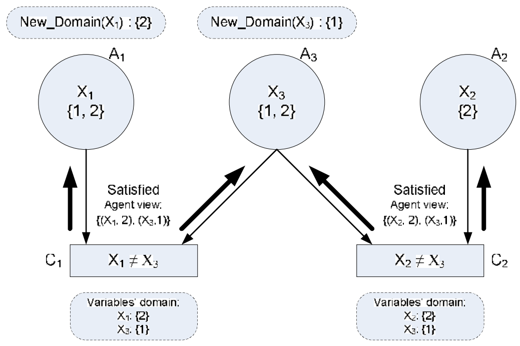

New_Domain(X1) : {2}
New_Domain(X3) : {1}
A1
A3
A2
X1 {1, 2}
X3 {1, 2}
X2 {2}
Satisfied Agent view: {(X1, 2), (X3,1)}
Satisfied Agent view: {(X2, 2), (X3,1)}
C1
X1 ≠ X3
X2 ≠ X3
C2
Variables' domain: X1: {2} X3: {1}
Variables' domain: X2: {2} X3: {1}

Figure 7 result of proposition validation

The domain reduction continues until we have stability in the variables' domain. The agents can start then proposing values for their variables. In Figure 6, agents A_1, A_2 and A_3 propose 2, 2 and 1 as values for their variables. They send these propositions to the related CAgents C_1 and C_2 for validation.

After validating the propositions, the CAgents send back the result of either to be satisfied or not (Figure 7).

3.5 Algorithm extension

As we said earlier in this paper, for simplicity reasons we will consider only the case where a VAgent has one variable only. In the same manner, we will treat the case where a CAgent contains one binary constraint only The extension of the proposed algorithm to be used with VAgents that have more than one variable is simple: If a VAgent VA_i has a set of interface variables that contains more than one variable $\{v_{i1}, \dots, v_{il}\}$ where each variable is involved in an external constraint, the agent should be related to all CAgents owning these constraints. The agent VA_i sends the domain of the variable v_{ij} only to the concerned controller(s) in order to avoid overloading communication. The rest of the manipulation stays the same as for an agent with one variable only.

Treating non-binary constraints is straight forward: the CAgent containing non-binary constraint exchange all necessary information with all VAgent having variables involved in its constraint. The same thing applies when the controller contains more than one constraint.

4 Framework implementation general structure

The general schema of the proposed framework is shown in Figure 8. The intended architecture represents a layer between the application and the layer of the real MAS and CSP platforms. From the application view point, the system is composed directly from the two principle types of agents: the CAgent and the VAgent. The user can create the necessary VAgents according to its problem definition. He also creates the constraints and associates them to CAgents.

The system layer uses generic interfaces for both MAS and CSP platforms. This allows the system to use any existed platforms by implementing these interfaces. In the same time this isolates the internal structure from the changes of choice of platforms. An intermediate layer between the system and the real MAS or CSP platform is necessary in order to separate the structure of the system from that of the real MAS and CSP platforms. This layer works as an adapter; it implements the generic platforms in the system layer using the real platforms. This implementation and its difficulty changes according to the MAS and CSP platforms used for the realization of the final system.

In our work we have chosen JADE [19] as a MAS platform. JADE (Java Agent DEvelopment Framework) is a software Framework compliant with the FIPA specifications and is fully implemented in Java language. It simplifies the implementation of multi-agent systems through a set of graphical tools that supports the debugging and deployment phases. JADE is free software developed and distributed by Telecom Italia Lab.

In which concern the CSP platform, we have chosen Choco. Choco [20] is a library for constraint satisfaction problems (CSP), constraint programming (CP) and explanation-based constraint solving (e-CP). It is built on an event-based propagation mechanism with backtrackable structures. It is an open-source software implemented in java also. It permits the use of multiple solvers for different problem separately. This allows each CAgent to have its own solver.

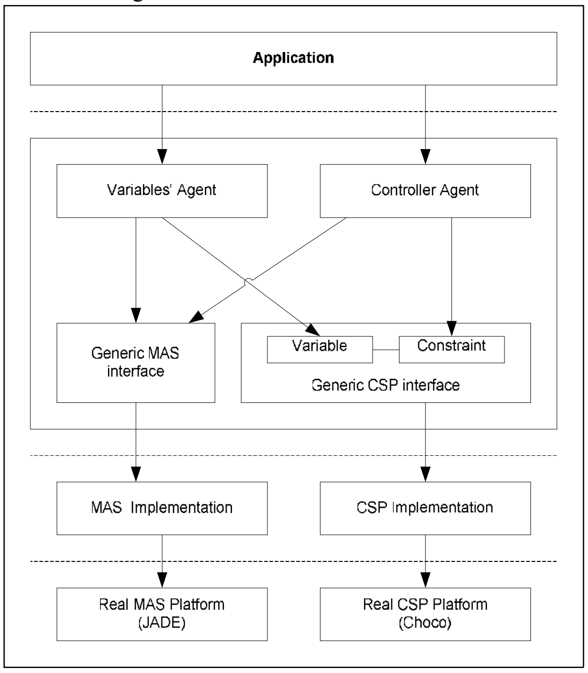


Figure 8 general schema of the proposed system

Both types of agent use the generic MAS platform that provides the base for constructing agents and manage the communication between them. CAgents use the generic CSP interface to have the capability of solving and validating constraints.

5 Conclusion and perspective

In this paper, we have introduced our different approach for the solution of Distributed Constraint Satisfaction Problems in a MAS consisting of two architectural types of agents: CAgents and VAgents. This approach allows the separation between the treatment of constraints and the other functionalities of the agents in the system. Also, it allows dividing a general constraint problem into multiple sub-problems easier to be treated.

Two aspects are investigated for enhancing the algorithm. We have the intention of enhance the algorithm role of the CAgent in two aspects: (a) by giving the CAgent the capacity of proposing solutions to the VAgents. These resolutions are established by negotiations with other CAgents, (b) by giving a preference level to every constraint. This allows constraints relaxing and permits CAgents to accept proposals which may not satisfy some weak constraints. This can be useful in case of over-constraint problems.

Another perspective is to guide the strategy of proposals of a VAgent by an optimization function. This function is defined according to the specific objectives to this agent.

References

[1] G. Ringwelski, "An Arc-Consistency Algorithm for Dynamic and Distributed Constraint Satisfaction Problems."

[2] C. Fernandez, R. Bejar, B. Krishnamachari, C. Gomes, and B. Selman, "Communication and Computation in Distributed CSP Algorithms."

[3] M. Yokoo and K. Hirayama, "Algorithms for Distributed Constraint Satisfaction: A Review," in Autonomous Agents and Multi-Agent Systems, 2000, pp. 198-212.

[4] C. Bessiere, I. Brito, A. Maestre, and P. Meseguer, "The Asynchronous Backtracking Family," LIRMM-CNRS, , Montpellier, France March 2003 2003.

[5] C. Bessiere and I. Brito, "Asynchronous Backtracking without Adding Links: A New Member in the ABT Family," 2005, pp. 7-24.

[6] M. Yokoo, "Asynchronous Weak-commitment Search for Solving Distributed Constraint Satisfaction Problems," in International Conference on Principles and Practice of Constraint Programming, 1995, pp. 88-102.

[7] C. Bessière, A. Maestre, and P. Meseguer, "Distributed Dynamic Backtracking."

[8] S. Al-Maqtari, H. Abdulrab, and A. Nosary, "A Hybrid System for Water Resources Management," in GIS third international conference & exhibition, 2004.

[9] F. Bacchus, X. Chen, P. v. Beek, and T. Walsh, "Binary vs. Non-Binary Constraints," 2002, pp. 1-37.

[10] F. Bacchus and P. v. Beek, "On the Conversion between Non-Binary and Binary Constraint Satisfaction Problems," in Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98), 1998, pp. 311-318.

[11] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara, "The Distributed Constraint Satisfaction Problem: Formalization and Algorithms," in IEEE Transaction on Knowledge and Data Engineering, 1998, pp. 673-685.

[12] M. Yokoo, T. Ishida, E. H. Durfee, and K. Kuwabara, "Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving," in 12th IEEE International Conference on Distributed Computing Systems, 1992, pp. 614-621.

[13] M. Yokoo, "Distributed Constraint Satisfaction Algorithm for Complex Local Problems," in Third International Conference on Multiagent Systems (ICMAS-98), 1998, pp. 372-379.

[14] R. Mailler and V. Lesser, "A Cooperative Mediation-Based Protocol for Dynamic, Distributed Resource Allocation," in IEEE Transaction on Systems, Man, and Cybernetics, Part C, Special Issue on Game-theoretic Analysis and Stochastic Simulation of Negotiation Agents, New York, 2004, pp. 438-445.

[15] H. Rudova, "Constraint Satisfaction with Preferences," in Faculty of Informatics Brno - Czech Republic: Masaryk University, 2001.

[16] S. Al-Maqtari, H. Abdulrab, and A. Nosary, "Constraint Programming and Multi-Agent System mixing approach for agricultural Decision Support System," in Emergent Properties in Natural and Artificial Dynamical Systems, 2006, pp. 199-213.

[17] P. Berlandier and B. Neveu, "Problem Partition and Solvers Coordination in Distributed Constraint Satisfaction," in Workshop on Parallel Processing in Articial Intelligence (PPAI-95), Montreal, Canada, 1995.

[18] R. H. Bisseling, J. l. Byrka, and S. Cerav-Erbas, "Partitioning a Call Graph."

[19] JADE: http://jade.tilab.com.

[20] Choco: http://choco.sourceforge.net/.

Introducing Bar Systems: A Class of Swarm Intelligence Optimization Algorithms

Esteve del Acebo and **Josep Lluis de la Rosa** [1]

Abstract. We present Bar Systems: a family of very simple algorithms for different classes of complex optimization problems in static and dynamic environments by means of reactive multi agent systems. Bar Systems are in the same line as other Swarm Intelligence algorithms; they are loosely inspired in the behavior a staff of bartenders can show while serving drinks to a crowd of customers in a bar or pub. We will see how Bar Systems can be applied to CONTS, a NP-hard scheduling problem, and how they achieve much better results than other greedy algorithms in the "nearest neighbor" style. We will also prove this framework to be general enough to be applied to other interesting optimization problems like generalized versions of flexible Open-shop, Job-shop and Flow-shop problems.

1 INTRODUCTION

The origin of the term Swarm Intelligence, which so vast amount of attention has drawn in the last years amongst the Artificial Intelligence, Artificial Life and Distributed Problem Solving communities is to be found in the observation of social insect colonies. A commonly accepted and used definition of it is: "the property of a system whereby the collective behaviors of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge". Doubtless, the paradigm of a Swarm Intelligence system is an ant colony. In it, individual ants' behavior is controlled by a small set of very simple rules, but their interactions (also very simple) with the environment allow them to solve complex problems (such as finding the shortest path from one point to another one). Ant colonies (and we could say the same about human beings) are intelligent systems with great problem solving capabilities, formed by a quantity of relatively independent and very simple subsystems which do not show individual intelligence. It is the"many dummies make a smart" phenomenon of emergent intelligence.

Swarm Intelligence problem solving techniques present several advantages over more traditional ones. On one hand, they are cheap, simple and robust; on the other hand, they provide a basis with which it is possible to explore collective (or distributed) problem solving without centralized control or the provision of a global model. Over the last years they have been used in the resolution of a very heterogeneous class of problems: Two of the most successful Swarm Intelligence techniques currently in use are Ant Colony Optimization [5] and Particle Swarm Optimization [8]. Ant Colony Optimization techniques, also known as Ant Systems, are based in ants' foraging behavior, and have been applied to problems ranging from determination of minimal paths in TSP-like problems to network traffic rerouting in busy telecommunications systems. Particle Swarm Optimization techniques, inspired in the way a flock of birds or a school of fish moves, are general global minimization techniques which deal with problems in which a best solution can be represented as a point or surface in an n-dimensional space. Other Swarm Intelligence applications include collective robotics , vehicle navigation, planetary mapping, streamlining of assembly lines in factories, coordinated robotic transport, banking data analysis and much more. The interested reader can find a lot of useful references about self-organization and Swarm Intelligence theory and applications in [1], [7], [9], [2], [6], and [3].

The class of systems we present in this paper, Bar Systems, are reactive multi agent systems whose behavior is loosely inspired in that of a staff of bartenders, and can be enclosed in the broader class of Swarm Intelligence systems,

The paper is organized as follows: in the next section we will present and formalize the concept of Bar System, in section 3 we present the CONTS problem, a NP-hard scheduling problem for multi agent systems which will serve us to test the performance of Bar Systems. In sections 4 and 5 we will see how to solve the CONTS using a Bar System and we will comment the results. Finally, in section 6, we will draw some conclusions and we will discuss some directions toward which future work can be directed.

2 BAR SYSTEMS

Anybody who has tried to get served a pint in a bar crowded with customers will have had more than enough time to wonder with boredom about the method used by waiters, if there is any, to decide which customer to pay attention to at each time. Sometimes there is not much point, to be served before, in having been waiting for long or in yelling at the waiter. Details like the bar area where the customer is, his/her sex, whether the waiter knows him/her or whether the waiter likes the customer's face determine to a high extent the way in which orders are served.

Let us examine the situation from the bartenders' point of view: a heap of customers are ordering drinks at once, new ones arrive all the time, and the bartenders have to do all they can to serve them. Of course, they cannot do it in an random way; they have to try to maximize some kind of utility function which will typically take into account aspects such as average serving time, average serving cost or average customer/boss satisfaction. They will have to pay attention, then, to facts such as that some of them can prepare certain drinks more quickly or better than others, that the order in which the drinks are served influences the time or the total cost of serving them, and that also moving from one place in the bar to another costs time. All of this without forgetting, on one hand, that the order in which orders

[1] Agents Research Lab. Institut d'Informtica i Aplicacions Universitat de Girona, Spain, email: acebo@ima.udg.es peplluis@eia.udg.es

take place has to be respected as much as possible and, on the other hand, that they have to try to favor the best customers by giving them special preferential attention and keeping them waiting for a shorter time.

The problem is not at all trivial, (actually we will see that it can be proved to be NP-hard), bartenders have to act in a highly dynamic, asynchronous and time-critical environment, and no obvious greedy strategy (such as serving first the best customer, serving first the nearest customer or serving first the customer who has arrived first) gives good results. Nevertheless, a staff of good bartenders usually can manage to serve a lot of customers in such a way that the vast majority of them were, more or less, satisfied. The way they accomplish the task seems to have little to do with any global planning or explicit coordination mechanisms but, arguably, with trying to maximize, every time they choose a customer to serve, some local utility function which takes into account aspects like the importance of the customer, the cost for the waiter of serving her/him and the time that he/she has been waiting for service.

In the next section, we will try to give a general formalization of this type of problem solving behaviors, which we call Bar Systems.

2.1 Definition

We will define a Bar System as a quadruple (E, T, A, F) where:

1. E is a (physical or virtual) environment. The state of the environment at each moment is determined by a set of state variables V_E. One of those variables is usually the time. We define S as the set of all possible states of the environment E, that is, the set of all the possible simultaneous instantiations of the set of state variables V_E.
2. $T = \{t_1, t_2, ..., t_M\}$ is a set of tasks to be accomplished by the agents within the environment E. Each task t_i has associated:
   - $pre(t_i)$. A set of preconditions over V_E which determine whether the task t_i can be done.
   - $imp(t_i)$. A nonnegative real value which reflects the importance of the task t_i.
   - $urg(t_i)$. A function of V_E which represents the urgency of task t_i in the current state of the environment E. It will be usually a nondecreasing function of time.
3. $A = \{a_1, a_2, ..., a_N\}$ is a set of agents situated into the environment E. Each agent a_i can have different problem-dependent properties (i.e. weight, speed, location, response time, maximum load...). For each agent a_i and each task t_j, $cost(a_i, t_j)$ reflects the cost for agent a_i to execute the task t_j in the current state of the environment. This cost can be divided in two parts: on one hand, the cost for a_i to make the environment fulfill the preconditions of task t_i (this can include the cost of stop doing his current task) and, on the other hand, the cost for a_i to actually execute t_j. If an agent a_i is unable to adapt the environment to the preconditions of the task t_j or if it is unable to carry the task out by itself then we define $cost(a_i, t_j)$ as infinite.
4. $F : S \times A \times T \rightarrow \Re$ is the function which reflects the degree to which agents are "attracted" by tasks. Given a state s of the environment, an agent a_i and a task t_j $F(s, a_i, t_j)$ must be defined in a way such that it increases with $imp(t_j)$ and $urg(t_j)$ and it decreases with $cost(a_i, t_j)$.

In Bar Systems, agents operate concurrently into the environment in a asynchronous manner, eliminating, thus, the typical operation cycles of other SI systems (Ant Systems, Particle Swarm Optimization Systems, Cellular Automata…). The general individual behavior of agents is given by the following algorithm:

```
REPEAT
 Find the most attractive free task MAFT;
 IF the agent is currently doing MAFT THEN
   keep doing it;
 ELSE
   Stop doing the current task, if any;
   IF pre(MAFT) hold THEN start doing MAFT
    ELSE do some action to fulfill pre(MAFT);
   ENDIF
 ENDIF
UNTIL no tasks left;
```

The crucial step in the algorithm above is the choice of the task which the agent has to execute for the next time step. In its simplest form, it can consist in choosing the one which maximizes the attraction function F . We will see in the next sections that it can also involve some kind of negotiation between agents and even some kind of local planning.

It is worth to stress the fact that the algorithm allows the agents to respond in real time to changes in the environment like the appearance of new urgent tasks or the temporal impossibility of fulfilling the set of preconditions of a given task.

2.1.1 *Inter-agent communication*

Even if Bar Systems don't require from the agents any communicative skills, they are indispensable in order for the system to attain the coordinated and self organized behavior typical of Swarm Intelligence Systems. We can identify three main purposes to which communication can serve in order to increase Bar Systems problem solving capabilities:

- *Conflict resolution and negotiation.* The way we defined Bar Systems makes unavoidable the occurrence of conflicting situations in which two or more agents choose the same task to carry out. Lack of communication will lead to a waste of resources because of several agents trying to fulfill the preconditions of the same task, even if only one of them will finally carry it out. In such situations it would be convenient to have some kind of negotiation method which can be as simple as "the first one which saw it goes for it". In the case study, in section 3, we will discuss a couple of more elaborated negotiation strategies.
- *Perception augmentation.* In the case that agents have limited perception capabilities (we refer to capability to perceive the tasks), communication can allow an agent to transmit to the others information about pending tasks they are not aware of. Let's suppose we want to do some kind of exploratory task in a vast terrain where points of interest must be identified and explored by means of a Bar System. It would be useful that agents had the ability to share information about the points of interest which they have located during their exploratory activity, this way agents would have access to information about the location of points of interest which lie beyond their perceptual capabilities.
- *Learning.* The attraction function f defined in section 2.1 does not need to be fixed in advance. Agents can learn it through their own activity and their communicative interactions with other agents. For example, an agent can find out that a certain kind of task has a high cost and communicate this fact to the other agents. Not only

that, agents can even learn from other agents the way of carrying out new tasks.

On the other side, It is worth to differentiate two main classes of inter-agent communicative processes:

- *Direct.* Agents establish direct communication with each other via some channel and following some kind of consensuated protocol.
- *Indirect.* Agents communicate with each other through their actions, which cause changes in the environment. In the Bar Systems framework, it can be seen as agents generating "communicative tasks" which, when carried out by other agents, increase the information they possess (about the environment, the task set . . .). This is the case of Ant Systems, which, from this point of view, can be seen as a particular case of Bar Systems.

2.1.2 Local planning

Although there is nothing like global planning in the way a set of bartenders work, they have tricks that allow them to spare time and effort. For example if two customers are asking for a pint and they are close enough to each other in the bar, the bartender will usually serve them at once. In a similar way, a taxi driver who is going to pick up a passenger will surely take advantage of the opportunity if he finds in his way a new passenger and he can transport him without deviating too much from his original route. The inclusion of this sort of very simple, problem-dependent, local planning techniques in the choice of the tasks is not difficult and can be done through different methods ranging from local search to the use of expert rules.

3 THE CONTS PROBLEM

A class of problems frequently found in "real life" involves some kind of scheduling in the transport of goods or people from one place to another. The problem which we present as a framework for the study of Bar Systems applicability and efficiency is inspired in the problem which has to be solved by a group of loading robots in a commercial harbor. The task of these robots is to transport the containers from their storage place to the docks where the corresponding ships have to be loaded. Of course, this transport has to be done in such a way that the containers arrive in time to be loaded and with the lowest possible cost. Next we state a formalization (and simplification) of the problem, which we will call CONTS. Afterward we are going to study its complexity and we will see how we can use a Bar System to solve it efficiently.

3.1 Definition of the problem

Let $C = \{c_1, c_2, ..., c_n\}$ be a set of containers, let $L = \{l_1, l_2, ..., l_m\}$ be a set of loading robots and let $P = \{(x, y) \in \{0..MaxX\} \times \{0..MaxY\}\}$ be a set of positions. Each container c_i has the following associated properties:

- $p(c_i) \in P$. The position where the container lies.
- $dest(c_i) \in P$. The position to which the container has to be carried to.
- $weight(c_i) \in \Re^+$. The weight of the container.
- $dline(c_i) \in \Re^+$. The latest instant of time in which the container can arrive to the dock in order to be loaded in time into the ship.

In order not to complicate the problem too much, we will assume that all the containers have the same importance. There are also several properties associated to each loading robot li:

- $p(l_i) \in P$. The place where the robot is at each instant.
- $maxload(l_i) \in \Re^+$. The maximum weight the robot is able to carry.
- $maxdist(l_i) \in \Re^+$. The distance beyond which the robot can't "hear". It allows us to model the perceptual limitations of the robot.
- $speed(l_i) \in \Re^+$. The speed at which the agent can move.

Robots can perform different actions, they can move toward any position, load (if container and robot are in the same position) containers which weigh less or the same as its $maxload$ value and download containers.

The problem consists in finding, if it exists, a sequence of actions that allows the robots, departing from their initial positions, to transport every container to its destination point, in such a way that no container arrives after its deadline. In order to simplify the problem, we will assume that the robots always move at the same speed, that uploading and downloading operations are instantaneous and that robots can only carry one container at a time.

3.2 Complexity of the CONTS problem

Of course, before trying to solve the problem we have to get an idea of its complexity. Using an heuristic method might not make much sense if there was some exact method of polynomial complexity. On the contrary, if the problem was very complex, using heuristic methods which gave approximate solutions, like Bar Systems, would be justified. The fact is that the problem is not at all trivial. The associated state space is enormous (it is not only necessary to take into account which containers each robot will move and in which order; the solution of some instances of the problem implies moving some containers to a different position from the one of delivery and leave them there to return to take them later) and it is also extremely sensitive to initial conditions, as most of NP-hard problems usually are. In [4] an in-depth study of the problem can be found with a proof of it to be at least as complex as a NP-hard problem. In general terms, the proof reduces the Traveling Salesman Problem (TSP) to CONTS by showing that every instance of the TSP problem is equivalent to an instance of CONTS where there is a single robot and all the containers have the same deadline and have to be delivered in the same position where they lie. We have also programmed an exhaustive search method that finds optimal solutions, but, as expected, it can only deal with extremely simple instances of the problem.

4 A BAR SYSTEM FOR SOLVING THE CONTS PROBLEM

Once the option of solving the problem in an exact way in the general case has been discarded, we now look at the possibility of using an heuristic method like a Bar System. The idea on which we are going to base it is very simple: to simulate an environment where the containers "shout" to the agents asking for somebody to take them to their destination. The intensity of the shout of each container depends on the remaining time before its deadline and the distance between its position and the delivery position (it could also depend on the importance of each container, but we must remember that the way we defined the problem, they are all equally important). The robots hear the calls of the containers diminished by the distance, so they go and take the ones they hear better. In order to achieve this behavior in the robots we will use a linear attraction function. Following the notation introduced in section 2, we define, for all container c and for all robot l, the attraction function F in the following way:

$$F(c,l) = \begin{cases} -\infty, & \text{if } c \text{ has been delivered.} \\ -\infty, & \text{if } c \text{ is being delivered for a} \\ & \text{robot other than } l. \\ K_1 \cdot urg(c) - K_2 \cdot cost(c,l), & \text{ow.} \end{cases} \quad (1)$$

Where K_1 and K_2 are adjustable parameters. The urgency function $urg(c)$ is defined as inversely proportional to the time which remains to c's deadline and takes into account the time required for transporting the container to its destination point:

$$urg(c) = curtime + \frac{d(p(c), dest(c))}{meanspeed} - dline(c) \quad (2)$$

Where d is the Euclidean distance, $curtime$ is the current time and $meanspeed$ is an environmental constant which averages agents' speeds. The $cost$ function is defined as follows:

$$cost(c,l) = \begin{cases} \infty, & \text{if } weight(c) \geq maxload(l). \\ \infty, & \text{if } d(p(l), p(c)) \geq maxdist(l). \\ \dfrac{d(p(l), p(c)) + d(p(c), dest(c))}{speed(l)}, & \text{ow.} \end{cases} \quad (3)$$

The election of this attraction function F is quite arbitrary. A nonlineal function would probably better reflect the "hearing" metaphor we talked about before. In the same way, we could also have defined a more sophisticated urgency function, non-linearly increasing depending on the time to the containers' deadline, for example. Bar Systems are general enough to use any attraction, cost or urgency functions. The question is finding, for each problem, the function which will give the best results. Our choice of the attraction function F is based in its simplicity, in spite of which, it has allowed us to obtain very good results.

The behavior of the robots will be very simple and it will obey the algorithm described in section 2.1. Each robot will choose a container to go for and will go toward its position, will load it (if not any other robot has arrived first) and will take it to the delivery point. After that, it will repeat the cycle until no containers left to transport.

4.1 Inter-agent communication and local planning for the CONTS problem

Aiming to the study of the utility of interagent communication, we will investigate two different methods for the choice of the next container to go for. If no communication between agents is allowed, each agent will simply choose the one which maximizes the attraction function. On the other hand, if the possibility of communication between agents is activated, each robot will ask to the others (perhaps not all of them but only those which communication is feasible) which containers they prefer and, in case of conflict (that is, another robot preferring the same container), a small negotiation process will start, the goal of which is to give preference to the agent who will be able to carry faster the container to its delivery position. The agent which finds itself in the situation where other agents have priority over it to transport its favorite container will try with the next best container, in order of preference according to its point of view, until if finds one for which it will have more priority than any other agent. It would be easy to devise more sophisticated negotiation processes taking into account the second-best options of the agents in conflict in such a way that one agent could resign carrying its preferred container, even if it has the higher preference over it, whenever the preference difference between the best and the second- best containers was small enough.

We have also implemented a very straightforward planning-like strategy in our Bar System. Whenever a robot has a container to go for, it looks if there exists another one such that it is possible to transport it without deviating too much from its original way to the first container position. If so, the agent transports it before resuming its original way to the first container position.

5 RESULTS

In order to analyze the efficiency of our method and experiment with different settings and parameter values, we have programmed a graphical simulator for the problem. We have chosen an instance of the problem with eighty containers randomly scattered on a 300×300 rectangular area with random delivery points and deadlines and four carrier robots, all of them with the same value for the parameter $maxdist$ and different speeds. We have done two main sets of simulations experimenting with different values of the parameters $K1$ and $K2$. In the first set (figure 1) we don't allow agents to communicate or perform any local planning, whereas in the second set (figure 2) communication and local planning are permitted.

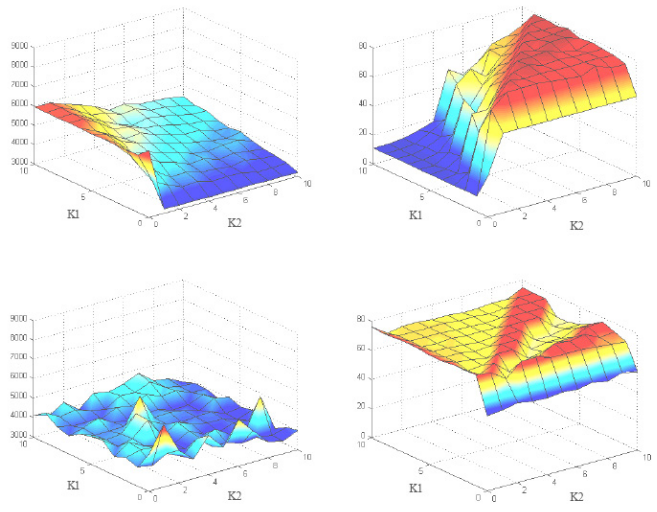


Figure 1. Left: Total time needed by the system to deliver all the containers for different values of the parameters $K1$ and $K2$ and for different values of the parameter $maxdist$ (top row $maxdist = 300$, bottom row $maxdist = 100$). Right: Number of containers delivered before their deadlines. Communication and local planning are deactivated.

We can see in figures 1 and 2 the results of the two sets of simulations. Each row represents a series of 121 simulations (for values of the $K1$ and $K2$ parameters ranging from 0 to 10 in increases of 1). The charts in the left columns show the time used to deliver all the containers and the charts in the right columns show the number of containers delivered before their deadlines. The two rows correspond to different values (300 and 100) of the $maxdist$ parameter.

We can draw several conclusions. On one hand, it is clear that, for some values of the parameters $K1$ and $K2$, the system finds much better solutions than those which can be obtained by using nearest neighbor-like methods. We can observe the performance of those methods in the top row of figure 1, when $K1 = 0$ the preference function F depends only on the $cost$ function and the systems

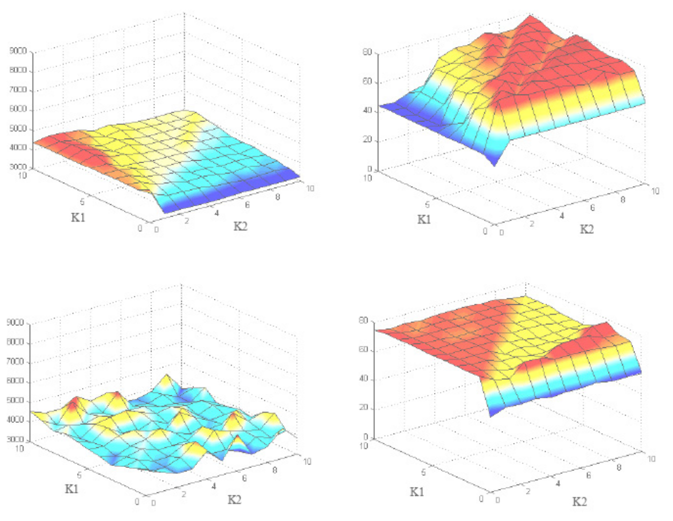


Figure 2. Left: Total time needed by the system to deliver all the containers for different values of the parameters $K1$ and $K2$ and for different values of the parameter $maxdist$ (top row $maxdist = 300$, bottom row $maxdist = 100$). Right: Number of containers delivered before their deadlines. Communication and local planning are permitted.

behaves in the "nearest container" way. The results are a low total delivery time and a considerable number of containers delivered after its deadline. The case $K2 = 0$ is even worse. The system follows the "most urgent container" behavior, resulting in very long displacements which cause a big total delivery time and ,consequently, a big number of containers delivered with retard. It is worth to remark that the improvement over those greedy methods achieved by our Bar System for some values of the parameters $K1$ and $K2$ is not attained in exchange of a greater complexity; in fact, the complexity of the system, understood as the amount of work which each agent has to do in order to decide the next container to go for, increases lineally with the number of containers.

We can also observe how the quality of the solutions found depends on the perceptual capabilities of the agents. When this capability is very limited (not shown in the figures), robots' behavior is too local, resembling somewhat like a mixture of"nearest container" and random walk. On the other side, very good solutions are found for certain values of the parameters $K1$ and $K2$ when the agents are able to perceive the environment almost entirely ($maxdist = 300$). This augmented perceptual ability implies, nevertheless, the possibility of appearance of several phenomena which can affect system's efficiency, like, for instance, that it will be necessary to evaluate more alternatives, that the probability of conflicts will increase and that, depending on the values of the parameters, the system can arrive to very bad solutions if the agents must perform long displacements. Thus, a bit paradoxically, more perception power can yield poorer results. The most interesting case, from our point of view, is when the agents have a perceptual capability between the two extreme points. We have tested the case $maxdist = 100$ and we can see in the bottom row of figure 1 how the system finds good solutions for most values of the parameters $K1$ and $K2$. There are particularly, two big zones in the parameters space where the solutions found are as good as the ones obtained by the agents of the first row of the figure, which have a perceptual power nine times greater.

In figure 2, we can see how the inter agent communication or negotiation and local planning can improve greatly, depending on the values of the parameters, the quality of the solutions found. Clearly, the importance of communication between agents increases with the possibility of conflict, which is proportional to the agents' perception power and decreases with the relative magnitude of the parameter $K2$ regarding $K1$.The more $K2$ grows regarding $K1$, the more importance is given to the distance to the container in the calculation of the preference function, the robots tend to prefer the nearest containers and the number of conflicts decrease, as well as the utility of communication.

It is interesting to note that for some values of the parameters $K1$ and $K2$, communication and local planning capabilities does not improve system's results. This is probably due to the fact that the results for those parameter values in the Bar System without communicative or planning capabilities are near-optimal (all the containers delivered in time). Nevertheless, it is clear that more work in this direction is needed in order to clarify communication and local planning effects.

6 CONCLUSIONS AND FUTURE WORK

We have presented Bar Systems, a new class of reactive multi-agent systems for distributed problem solving. They are loosely inspired in the way a group of waiters work behind a bar. We have formalized the definition and we have given the general algorithm which rules the behavior of the individual agents. Several of the characteristics of Bar Systems are:

- Simplicity. Agents in Bar Systems are simple. They share a similar structure and operate in a decentralized manner in a similar way optimizing a local function. Interactions between agents are simple, too.
- Efficiency. Bar Systems have lineal complexity with respect to the number of tasks.
- Robustness. Faults in individual agents do not decrease dramatically the efficiency of the system. Moreover, Bar Systems' problem solving capabilities increase steadily with the addition of new agents.
- Responsiveness. Bar Systems respond easily to the occurrence of unforeseen events as the appearance of new high priority tasks.

All those characteristics, jointly with the capability to seamless integrate different more or less sophisticated negotiation and local planning techniques, make Bar Systems very suitable to solve problems in asynchronous, dynamical, partial information and time-critical environments.

To check the efficiency and applicability of Bar Systems, we have defined a NP-Hard problem called CONTS, based on the work which a set of robots has to perform to transport a set of containers in time to their destination. The Bar System used to solve it has proved to give much better results than other greedy algorithms of the nearest neighbor type and has established, in our opinion, the usefulness of Bar Systems as a general framework for solving this type of real time problems. We have also seen that communication amongst agents and local planning allows improving the results greatly without increasing the complexity of the system significantly.

Our work in Bar Systems is just starting and we are aware that there are many aspects that require more study and testing. Some of the directions in which it would doubtless be worth working and which essentially refer to the nature of the attraction function F are:

- Study of more sophisticated negotiation strategies in case of conflict (two agents preferring the same task) and new local planning operators and its impact in system's performance.

- Study of Bar Systems' performance in highly dynamical, time-critical environments. We are currently considering its use in the ROBOCUP and RESCUE environments.
- Study of the applicability of Bar Systems to other kinds of problems. At a first glance it could seem Bar Systems to be a bit too restrictive with respect to the kind of problems which they can tackle, It must be remarked, nevertheless, that its application is not limited to problems involving the transportation of goods or people (and we don't mean it to be a narrow application field, it is wide enough to contain problems ranging from service assignation in cab companies to multi-agent autonomous terrain exploration), they can also be useful in other problems which do not necessarily involve physical movement of the agents or goods transportation, such as resource allocation problems in the style of flexible Open-Shop problems where the order in which a set of machines, in a factory, for instance, has to perform a set of operations has to be decided. In this type of problems, machines would correspond to the robots in the CONTS problem, tasks would correspond to containers transportations and the preconditions and postconditions for the tasks would correspond to the initial and destination positions of the containers in the yard. Moreover, with an appropriate definition of the attraction function F, a Bar System can be used for solving flexible Job-Shop problems, where there is a set of independent jobs, each formed by a series of tasks which have to be done in a specific order. In this kind of problems, for each job there is at all times, at the most, just one feasible task, and it would be sufficient to define the attraction functions in such a way that all job's not done tasks"transmit" their urgency to the feasible one. The same idea could be used in a more general setting, where there would simply be any type of non-cyclical precedence relations over the set of tasks. It can also be worth to study the applicability of Bar Systems in competitive environments.

REFERENCES

[1] Holland O.E. Beckers, R. and Deneubourg J.L., *From Local Actions to Global Tasks: Stigmergy in Collective Robotics*, 181–189, Artificial Life IV, MIT Press, Cambridge, MA, 1994.

[2] E. Bonabeau, M. Dorigo, and G. Thraulaz, *Swarm Intelligence. From Natura to Artificial Systems*, Oxford University Press, 1st edn., 1999.

[3] E. Bonabeau and G Thraulaz, 'Swarm smarts', *Scientific American*, **March 2000**, 72–79, (2000).

[4] E. del Acebo, 'Agents, sistemes multiagent i soluci distribuida de problemes', *Research Report. Institut d'Informatica i Aplicacions. Universitat de Girona*, (2005).

[5] M. Dorigo and T. Sttzle, *Ant Colony Optimization*, MIT Press, 2004.

[6] P. Engelbrecht A., *Fundamentals in Computational Swarm Intelligence*, John Wiley and Sons, 2006.

[7] M. A. Lewis and G. A. Bekey, *The Behavioral Self-Organization of Nanorobots Using Local Rules*, Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1992.

[8] K.E. Parsopoulos and M.N. Vrahatis, 'Recent approaches to global optimization problems through particle swarm optimization', *Neural Computing*, **1 (2-3)**, 235–306, (2002).

[9] M. Resnick, *Turtles, Termites and Traffic Jams, Explorations in Massively Parallel Microworlds*, MIT Press, 1997.

A Comparison between GAs and PSO in Training ANN to Model the TE Chemical Process Reactor

Malik Braik and **Alaa Sheta** and **Amani Arieqat**[1]

Abstract. In this paper, the adaptation of network weights using Particle Swarm Optimization (PSO) was proposed as a mechanism to improve the performance of Artificial Neural Network (ANN) in modeling a chemical process. This is particularly useful for cases involving changing operating conditions as well as highly nonlinear processes. As a case study, a Tennessee Eastman (TE) chemical process reactor was considered. Four subsystems of the reactor were considered. They are the reactor level, the reactor pressure, the reactor cooling water temperature, and the reactor temperature. PSO is proposed to allow automatic update of network weights to increase the adaptability to dynamic environment. Comparisons were also made to training the ANN using Genetic Algorithms (GAs). The results obtained in this paper confirmed the potential of PSO-based ANN model to successfully model the TE process. The results are explored with a discussion using the Mean Square Error (MSE) and Variance-Account-For (VAF) to illustrate the usability of the proposed approach. Finally, conclusions and future works are derived.

1 Introduction

Over the years, the application of Artificial Neural Network (ANN) in process industries has been growing in acceptance. Today, there is a growing interest of using ANNs to assist building a reasonable model structure for physical nonlinear systems [25]. ANNs have a special capacity to approximate the dynamics of nonlinear systems in many applications in a black box manner. An example of dynamical nonlinear systems is a Tennessee Eastman (TE) chemical reactor process [6]. Given sufficient input-output data, ANN is able to approximate any continuous function to arbitrary accuracy. In general, the development of a good ANN model depends on several operators. The first operator is related to the input-output data driven, where model qualities are mostly influenced by the quality of data being used. The second operator is concerning with the network architecture. Different network architectures result in a different estimation performance. The third operator is the model size and its complexity. Where a small network may be not able to represent the real situation of the model estimation due to its limited capability, while a large network may has noise in the training data and hence fail to provide good generalization ability, and the last operator is related to the quality of the process model, and is strongly dependent on the network training. May be the last issue, is the most important among all, since it is essentially an identification of model parameters that fit with the given data. The work in this paper will focus on the last issue.

Until today, many researchers prefer of using gradient descent algorithms such as Back-Propagation (BP) method. Some of the advantages of this gradient-based technique include its efficient implementations, good at fine-tuning, and faster convergence when compared with other methods. However, these methods are subject to problems involving local minima-since they are local search methods and when applied to complex nonlinear optimization problems, may be sometimes result in unexpected performances. These gradient methods assess the error in the network's decision as compared to a supervisor, and propagate the error to the weights throughout the network, so that, one of the main obstacles due to the fact that searching of optimal weights is strongly dependent on initial weights, and if they are located near local minima, the algorithm would be stuck at a sub-optimal solution. So that, the conventional gradient search method is susceptible to be converged at local optima. This is however not the case for Evolutionary Algorithms (EAs) since the genetic search methods offer better chances to get to the global optima. Furthermore, since EAs does not use gradient information, it is advantageous for problems where such information is unavailable or very costly to obtain. These advantages make them more robust and appealing than many other search algorithms [7].

Several different attempts have been proposed by various researchers to propitiate this training problem. These include imposing constraints on the search space, restarting training at many random points, adjusting training parameters, and restructuring the ANN structure [25]. One of the most promising techniques is by introducing adaptation of network training using (EAs) such as Particle Swarm Optimization (PSO), and Genetic Algorithms (GAs). Unlike BP, PSO is a global search algorithm based on the principle "survival of fittest". PSO is quite suitable for problems with many local minima or problems where gradient information isn't readily available, PSO avoid trapping in a local minima, because it is not based on gradient information [1, 15].

PSO has been used to train neural networks, finds neural network architectures, tunes network learning parameters, and optimizes network weights. The global best or local best solution in PSO is only reported to the other particles in a swarm. Therefore, evolution only looks for the best solution and the swarm tends to converge to the best solution quickly and efficiently, thus increasing the probability of global convergence. In this way, the merging of EAs and ANN will gain adaptability to dynamic environment and lead to significantly better intelligent systems than relying on ANN, PSO, or GA alone. In [20], Montana and Davis reported the successful application of a GA to a relatively large ANN problem. They proved that GA produce results superior than BP method. In [32], Yao and Liu presented a new evolutionary algorithm, to evolve ANN architecture and connection weights simultaneously. When tested on a number of

[1] Information Technology Department, Prince Abdullah Bin Ghazi Faculty of Science and Information Technology, Al-Balqa Applied University, Jordan, email:{m_fjo,asheta2,amany_arieqat}@yahoo.com

benchmark problems such as medical diagnosis problems, and credit card assessment problems. In [34], Zuo and Wu used GA to determine the optimal feeding rate for a hybrid ANN model of a fermentation system.

In other words, the developed models using ANN-EAs should be more robust to dynamic nonlinear process system [33]. In this paper, GAs-based ANN and PSO-based ANN are introduced for connection weights in ANN modeling.

Following this introductory section, the rest of the paper is organized as follows: In Section II, a background of the case study is presented. In section III: the preparatory works for ANN model development such as sensitivity analysis, and model input selection are reported. Subsequently, data collection and scaling procedure for model development are described. Also, procedures of ANN model development have been discussed. This is followed in section IV: by discussion of PSO and GAs and the motivation to include evolution in ANN modeling using EAs, also, the research methodologies involved are described. In section V, The reported results are simulated and discussed, and finally, the paper closes with some concluding remarks and future research directions in Section VI.

2 Tennessee Eastman Chemical Process

Over the years, the processes involved in manufacture and production of purified chemicals have become increasingly more complex. Systems that were once controlled by operators based on the system performance have been converted to automatic control, where a mechanical or electrical controller adjusts valves and pumps [4, 17, 29]. Engineers have learned that, to develop the control algorithms for large complex systems, it is important to first create and test a model of the system offline. There are two reasons to do this in chemical plants, they are:

- Protecting humans and the environment from unsafe conditions.
- Shorten the design and analysis of the system [31].

2.1 Features of the TE Process

TE process, as presented by Downs and Vogel in [6], is based on an actual system, with slight changes made to protect the identity of the reactants and products. The system is a wellposed problem for analysis and control design of a nonlinear, open-loop unstable chemical process. The plant consists of five major operations: a two phase reactor, a product condenser, a vapor/liquid separator, a recycle compressor, and a product stripper. The nonlinear dynamics of the plant are mainly due to the chemical reactions within the reactor. A Nonlinear Predictive Control shown in Figure 1 was simulated [19]. The TE reactor process shown in this figure is borrowed from [3].

Downs and Vogel [6] proposed the following control objectives for this system:

1. Maintain process variables at desired values.
2. Keep process operating conditions within equipment constraints.
3. Minimize variability of product rate and product quality during disturbances.
4. Minimize movement of the gas valves which affect other processes.
5. Recover quickly and smoothly from disturbances, production rate changes, or product mix changes.

A suitable controller must be able to achieve at least these control objectives.

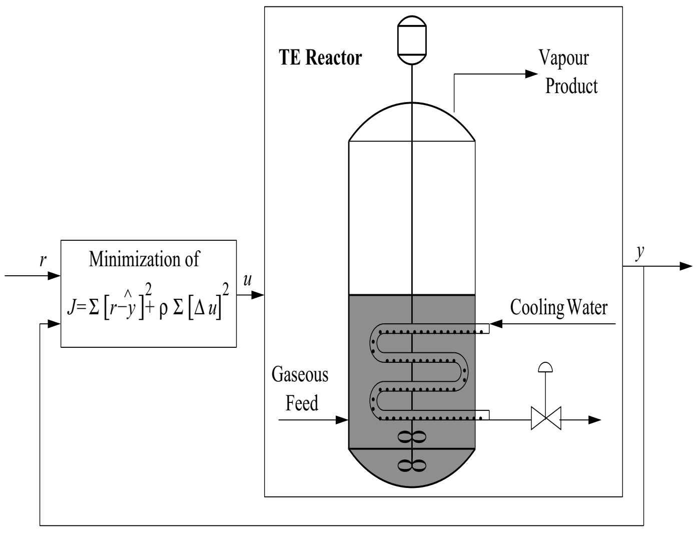


Figure 1. The Simulated Predictive Control Principle

2.2 Inputs and Outputs of the TE System

Typically, any chemical system can be represented by a set of mathematical equations that map the inputs and outputs of the system with a set of state variables. The main set of equations for normal operation of the chemical processes is the dynamic of equations associated with the reactions, unit operations, and flows of the system. The plant model has 12 manipulated variables, 50 states, and 41 measurements from different locations in the plant [12]. The 20 disturbance scenarios can be turned on and off to test the robustness of the system. Of the measured values, there are 22 values that monitored continuously. The other 19 measured values are retrieved from gas chromatographs (analyzers) that give information only at certain intervals. The three analyzers give information about chemical components in the reactor feed, the purge gas, and the product with sampling delays of 0.1 hour, 0.1 hour and 0.25 hour, respectively. The 50 states of the model closely correspond to the actual plant operation that represents the significant system dynamics. Fast dynamics, such as transmitter lags, are modeled as straight gains because the effects of these transients are minimal. The 12 manipulated variables in the TE process allow the designer 12 degrees of freedom with 9 flow valves, 2 temperature control valves, and control of the reactor agitation rate [12, 11].

3 Preparation for Model Development

Before the ANN model can be selected, some preparation stages must be completed. These tasks included: data collection, model structure selection. These preparations are necessary to provide model development.

3.1 Data Collection and Description

Dataset for training and evaluating is collected from various operating conditions to measure different outputs of the TE reactor. A total of 300 measurements were downloaded from [23]. The measurements describe the behavior of the TE reactor and show how it responds to various inputs. The dataset is split into two parts; the training dataset which is used to train the NN model, whilst a testing dataset is used to verify the accuracy of the trained NN model. The training dataset consists of 50% of the total dataset with the testing dataset consisting of the remaining 50%.

Since the model was built using time-series data, measurements at past sampling time $(t-1)$ for input variables were also included as model input. This was due to the fact that chemical process variables were always auto-correlated. Perhaps, by including the delay measurement as a model input for current estimation, the generalization ability of a feedforward network would be improved, and hence, the performance of the network in estimating the TE process would be enhanced.

The TE reactor modeling problem is divided into four sub-problems. They are: The reactor level, the reactor pressure, the reactor cooling water temperature, and the reactor temperature. Each of the four sub-problems has four input variables [2, 29], they are:

- $u_1(k)$ stand for the flow to the reactor.
- $u_2(k)$ stand for the coolant valve position,
- $u_3(k)$ stand for the feed mole fraction, and
- the fourth input $y(k-1)$ represents the past output variable value. The output for each sub-problem is represented by $y(k)$.

3.2 Data Scaling

Dataset is not normally used directly in process modeling of ANN. This is due to the difference in magnitude of the process variables. The data was scaled to a fixed range to prevent unnecessary domination of certain variables, and to prevent data with larger magnitude from overriding the smaller and impede the premature learning process. The choice of range depends on transfer function of the output nodes in ANN. Typically, $[0,1]$ for sigmoid function and $[-1,1]$ for hyperbolic tangent function. However, due to nonlinear transfer function has asymptotic limits; the range of dataset is always set slightly less than the lower and upper limits. In this work, since the sigmoid function is adopted, the data is normalized in the range of $[0.1-0.9]$ using the relation in equation 1:

$$v_i^{norm} = \frac{0.8(v_i - v_i^{min})}{(v_i^{max} - v_i^{min})} \tag{1}$$

where v_i is the input variables i or the output variable to be scaled. v_i^{min} and v_i^{max} are the smallest and the largest value of the data. The choice of $[0.1-0.9]$ was recommended based on the facts that it eliminates some of the saturation region at both end of the sigmoid function [9].

3.3 Model Structure Selection

Selecting a model structure from a set of candidate models is a difficult problem since the TE is a nonlinear process [22, 24]. Architecture of ANN model includes type of network, number of input and output neurons, transfer function, number of hidden layers as well as number of hidden neurons. Normally, the input neurons and output neurons are problem specific. In this work, Multi-Input Single-Output (MISO) structure had been utilized to estimate the TE process; hence, there will be only one output neuron. Since the model was built using time-series data, measurements at past sampling time $(t-1)$ for input variables were also included as model input. This was due to the fact that chemical process variables were always auto-correlated. In this way, the performance of the network in estimating the TE process would be enhanced. Thus, together with current sampling time for 3 input variables, there were 8 input neurons in the input layer. In this work, the TE reactor has only one hidden layer and 7 neurons in the hidden layer. It was utilized that an ANN with one hidden layer was sufficient in performing process mapping. Only one hidden layer with 7 neurons in the hidden layer was utilized in this work as it had proved that an ANN with one hidden layer was sufficient in performing process mapping arbitrarily [10]. From the optimum network topology, it was shown that the ANN model does not need a large number of hidden neurons. This revealed that the estimation task was not too complicated.

Also, it was important that the transfer function possessed the properties of continuity and differentiability. Commonly, log sigmoid function was utilized in the hidden layer and the output generated had a value between 0 and 1. However, the linear transfer function was more suitable in output layer. The equations for the log and linear transfer functions used in this work are shown in equations 2 and 3:

$$f(x) = \frac{1}{1+exp(-x)} \tag{2}$$

$$f(x) = x \tag{3}$$

4 Genetic Algorithms (GAs)

GAs belongs to a class of population-based stochastic search algorithm that is inspired from principles of natural evolution known as Evolutionary Algorithms [5]. GA is based on the principle of "survival of fittest", as in the natural phenomena of genetic inheritance and Darwinian strife for survival. GA operates on a population of individuals which represent potential solutions to a given problem. Imitating the biological principles in nature, a single individual of a population usually is affected by other individuals in its environment. Normally, the better an individual performs under these competitive conditions, has a greater chance to survive and reproduce. This in turn inherits the good parental genetic information. Hence, after several generations, the bad individual will be eliminated and better individuals will be produced. In general, GA is applicable to a wide range of optimization problems. Primarily, GA was designed to optimally solve sequential decision processes more than to perform function optimization but over the years, it has been used widely in both learning and optimization problems [30, 28]. There are two important issues in searching strategies for optimization problems: exploiting the best solution and exploring the search space [8, 27]. GA makes a balance between the exploitation and exploration of the search space. It allows the exploration of all solution space, which may reduce the converging to a local minimum. The exploitation in the neighborhood of possible solutions will perform as the high fittest solutions to be developed. Basically, the performance of weights evolution using GA depended on the number of populations and generations. If these parameters were set too low, the evolution may converge to immature solution. However, the larger number of populations and generations would require longer computation time for convergence [26].

In general, GAs used four steps to obtain the optimum connection weights of ANN:

step 1 Generate an initial population of random weights and the corresponding ANN was constructed with those weights.

step 2 ANN was evaluated using the population weights. This was done by computing its training error and assigning it to a fitness value according to how good the solutions are.

step 3 Parents for genetic manipulation were selected and a new population of weights were created.

step i The best existing weights (reproduction) were copied.

step ii New weights were created by crossover and mutation operators.

step 4 The best population of weights that appeared in any generation was designated as the result of the performed GA, for the weights evolved using GA, the number of generation was used to stop the iteration.
In this work, 50 populations of weights were evolved for 50 generations. The performances in the validation sets were considered in the acceptable level; this proved that this scheme was adequate with a sufficient accuracy.

5 Particle Swarm Optimization (PSO)

PSO is a global optimization technique that has been developed by Eberhart and Kennedy in 1995 [13, 14], the underlying motivation of PSO algorithm was the social behavior observable in nature, such as flocks of birds and schools of fish in order to guide swarms of particles towards the most promising regions of the search space. PSO exhibits a good performance in finding solutions to static optimization problems. It exploits a population of individuals to synchronously probe promising regions of the search space. In this context, the population is called a swarm and the individuals (i.e. the search points) are referred to as particles. Each particle in the swarm represents a candidate solution to the optimization problem. In a PSO system, each particle moves with an adaptable velocity through the search space, adjusting its position in the search space according to own experience and that of neighboring particles, then it retains a memory of the best position it ever encountered, a particle therefore makes use of the best position encountered by itself and the best position of neighbors to position itself towards the global minimum. The effect is that particles ”fly” towards the global minimum, while still searching a wide area around the best solution [15, 21, 16]. The performance of each particle (i.e. the ”closeness” of a particle to the global minimum) is measured according to a predefined fitness function which is related to the problem being solved. For the purposes of this research, a particle represents the weight vector of NNs, including biases. The dimension of the search space is therefore the total number of weights and biases.

The iterative approach of PSO can be described by the following steps:

step 1 Initialize a population size, positions and velocities of agents, and the number of weights and biases.

step 2 The current best fitness achieved by particle *p* is set as *pbest*. The *pbest* with best value is set as *gbest* and this value is stored.

step 3 Evaluate the desired optimization fitness function *fp* for each particle as the Mean Square Error (MSE) over a given data set.

step 4 Compare the evaluated fitness value *fp* of each particle with its *pbest* value. If $fp < pbest$ then $pbest = fp$ and $best_{xp} = xp$, xp is the current coordinates of particle p, and $best_{xp}$ is the coordinates corresponding to particle *p's* best fitness so far.

step 5 The objective function value is calculated for new positions of each particle. If a better position is achieved by an agent, *pbest* value is replaced by the current value. As in Step 1, *gbest* value is selected among *pbest* values. If the new *gbest* value is better than previous *gbest* value, the *gbest* value is replaced by the current *gbest* value and this value is stored. if $fp < gbest$ then $gbest = p$, where *gbest* is the particle having the overall best fitness over all particles in the swarm.

step 6 Change the velocity and location of the particle according to Equations 4 and 5, respectively [13, 18].

step 7 Fly each particle p according to Equation 5.

step 8 If the maximum number of a predetermined iterations (epochs) is exceeded, then stop; otherwise Loop to step 3 until convergence. In this work, 25 populations of weights were evolved for 200 generations.

$$\begin{aligned} V_i = wV_{i-1} \quad &+ \quad acc * rand() * (best_{xp} - xp) \\ &+ \quad acc * rand() * (best_{xgbest} - xp) \end{aligned} \tag{4}$$

Where *acc* is the acceleration constant that controls how far particles fly from one another, and *rand* returns a uniform random number between 0 and 1.

$$xp = xpp + V_i \tag{5}$$

V_i is the current velocity, V_{i-1} is the previous velocity, xp is the present location of the particle, xpp is the previous location of the particle, and i is the particle index. In step 5 the coordinates $best_{xp}$ and $best_{xgbest}$ are used to pull the particles towards the global minimum.

5.1 Tuning Parameters for GAs and PSO

To develop an accurate process model using ANN, the training, and validation processes are among the important steps. In the training process, a set of input-output patterns is repeated to the ANN. From that, weights of all the interconnections between neurons are adjusted until the specified input yields the desired output. Through these activities, the ANN learns the correct input-output response behavior. The model training stage includes choosing a criterion of fit (MSE) and an iterative search algorithm to find the network parameters that minimizes the criterion. PSO as well as GAs are used in an effort to formalize a systematic approach to training ANN, and to insure creation of a valid model. They are used to perform global search algorithms to update the weights and biases of neural network. The combinations of control parameters used for running PSO and GAs are shown in Table 1and Table 2 respectively:

Table 1. The Control Parameters used for Running PSO

Parameter	Value
Number of population	50
Number of generations	2000
Learning factors	1.3
Inertia weights	0.6 and 0.9
Fitness	MSE

Table 2. The Control Parameters used for Running GAs

Parameter	Value
Number of population	50
Number of generations	2000
Crossover probability	0.9
Mutation probability	0.03
Selection function	Ranking
Fitness	MSE

For the validation process, the expected model is compared graphically with the behavior of the target model.

6 Training and Validation Results

The performance of the proposed models in tracking the actual process data during the training and validation testing stages of the reactor level is illustrated in Figures 2,3, respectively. The modeling process of the reactor Level is shown as a sample, in overall, all estimator models display good performance in the training and validation sets indicating that the models developed are able to represent the behavior of the TE process.

The final convergence value of the proposed models reached 0. It can be seen from Figure 4 that the convergence process results in smooth curves, with a rapid increase at the start that gradually slows down. The experiments were implemented five times to ensure that MSE converges to a minimum value. In the reactor level problem, the MSE error converges to a value of 0.00034721 using PSO, and to 0.00038193 using GAs, while in the reactor pressure problem, the MSE converges to a value of 0.0001550 using PSO, and to 0.0001705 using GAs. In the other models, the MSE convergence value is nearly 0.

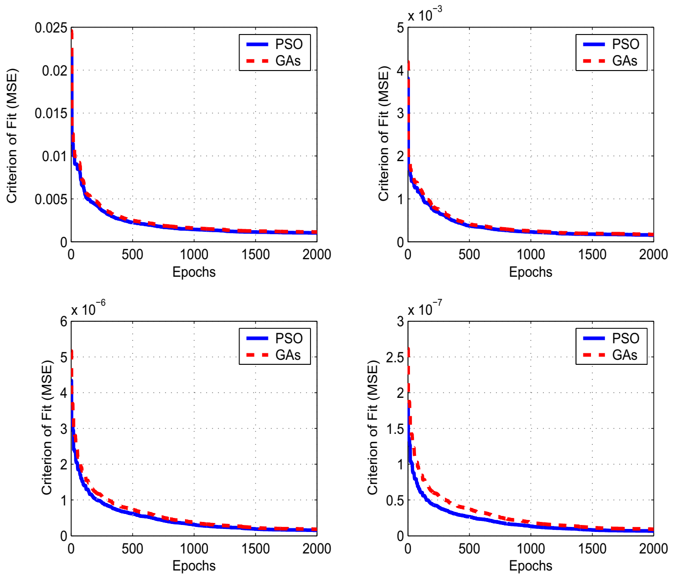


Figure 4. Convergence Curves of TE sub-problems; upper left-Reactor Level; upper right-Reactor Pressure; lower left-Reactor Cooling Temperature; lower right-Reactor Temperature

6.1 Evaluation Criteria

The performance of the proposed approach is evaluated by measuring the estimation accuracy. The estimation accuracy can be defined as the difference between the actual and estimated values. The first typical fitting criterion (MSE) is defined as in Equation 6:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad (6)$$

where N is the total number of data, y is actual target value, and $\hat{y}$ its estimated target value.

Since the calculation started with random initial weights, each run produced different results even though the network architecture was maintained. Thus, in order to obtain an optimal solution, tedious approaches such as repeating runs for the same topology cannot be avoided. The experiments are implemented many times to ensure that MSE converges to a minimum value. It was found that the TE reactor composition could be best estimated using a network with 7 hidden neurons. Using this network topology, the training and validation errors of the models developed based MSE are tabulated in Table 3.

Table 3. Training and Validation Results of The Estimator Models

problem	Train-PSO	Valid-PSO	Train-GAs	Valid-GAs
1	1.9321e-1	3.7736e-2	2.0820e-1	0.8226e-1
2	4.2778e-3	4.0076e-3	6.5994e-3	5.1018e-3
3	3.3147e-6	7.6189e-7	1.0179e-4	2.2881e-5
4	5.1391e-6	6.0090e-6	4.7434e-5	5.3340e-5

The second criterion fit, Variance-Account-For (VAF) is used to test the ability of the proposed approach to model the TE reactor in recall or testing phase, VAF is given in Equation 7.

$$VAF = 1 - \frac{var(y - \hat{y})}{var(y)} \times 100\% \qquad (7)$$

where, $y(t)$ is the actual system output, and $\hat{y}(t)$ is the predicted ANN output.

The results of modeling the TE reactor using ANN-PSO and ANN-GAs based on the VAF criterion are computed and reported in Table 4.

Table 4. VAF Values of the Proposed Models

Sub-problem	ANNs-PSO	ANNs-GAs
Reactor Level	90.9445	87.3228
Reactor Pressure	91.9994	89.8116
Reactor Cooling Temperature	99.7988	98.3863
Reactor Temperature	99.9944	99.7300

From the results shown in Tables 3, 4, when the comparison of training and validation performances was made between ANN-PSO and ANN-GAs model, ANN-PSO can perform better, meaning that PSO is very effective in training the NNs, learned to successfully navigate the course on the majority of test runs, and often reached an optimal MSE. ANN-GAs is not effective in modeling TE process; this is due to that population of weights for ANN-GA may not be able to reach a global optimum when the evolution was simulated over the same number of generations as with ANN-PSO model. The higher number of generations may be used but this is not recommended due to longer convergence time.

7 Conclusions and Future Work

This work has proposed an enhancement to neural network model. Aiming to improve the model robustness, evolution in network connection weights using EAs was explored. Based on the results obtained in this study, the main conclusions of this paper are: ANN is an efficient and effective empirical modeling tool for estimating the chemical process variable by using other easily available process measurements and the use of multilayer feedforward network with delay values in model input variables are sufficient to give estimation to any arbitrary accuracy. Also, this paper has taken advantage

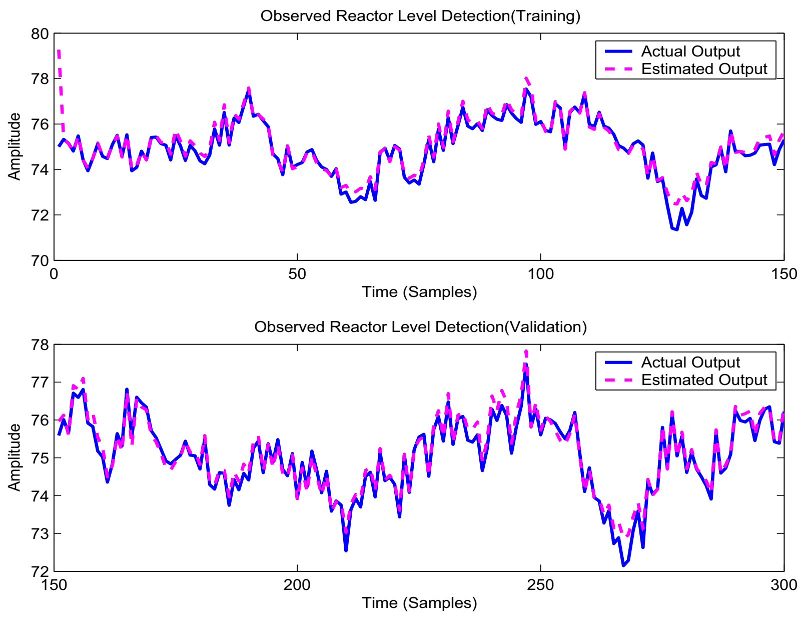


Figure 2. Observed Reactor Level: Training and Validation Performance of ANN-PSO Model

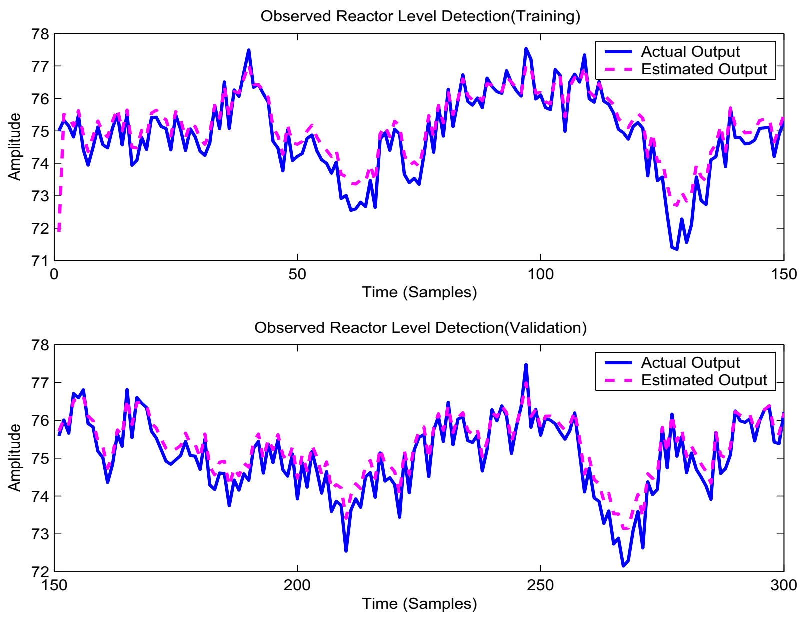


Figure 3. Observed Reactor Level: Training and Validation Performance of ANN-GA Model

of flexibility EAs techniques by applying it to the problem of training neural networks. In particular the global optimization method PSO is employed to provide a sense of the directivities of optimization the weights and biases of neural networks, and seek a good starting weight vector for subsequent neural networks learning algorithm. The preliminary results give a positive indication of the potential offered by EAs; this ensures the ability to effectively train the neural networks using the optimization techniques. In addition, PSO offered an increased level of adaptability of neural networks and is more preferable as the optimal solution searching to model a TE reactor problem than GAs. Despite the encouraging finding was obtained, there are still several further works to be considered. These include: The inclusion of adaptive feature using EAs to improve model robustness can be extended to evolution of a network architecture, which is typically number of hidden nodes as well as number of hidden layers; also the evolution of network architecture requires new set of connection weights.

Acknowledgements

Authors would like to acknowledge the financial support of Al-Balqa Applied University, Al-Salt, Jordan.

REFERENCES

[1] Hussein A. Abbass, Ruhul Sarker, and Charles Newton, 'PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems', in *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, volume 2, pp. 971–978, Piscataway, New Jersey, (May 2001). IEEE Service Center.

[2] Heba Al-Hiary and Alaa Sheta, 'A new neural networks model for industrial processes', in *4th International Multiconference on Computer Science and Information Technology CSIT*, volume 2, pp. 189–198, (2006).

[3] I.Paul Barton and S.Wade Martinsen, 'Eqaution-oriented simulator training', in *Proceedings of the American Control Conference, Albuquerque, New Mexico*, pp. 2960–2965, (1997).

[4] N. Bhat and T. J. McAvoy, 'Use of neural nets for dynamic modelling and control of chemical process systems', *Computers & Chemical Engineering*, **14**(4), 573–583, (1990).

[5] M. Choenauer and Z. Michalewicz, 'Evolutionary computation control and cybernetics', *Proceedings of the IEEE*, **26**(3), 307–338, (1997).

[6] J. J. Downs and E. F. Vogel, 'A plant-wide industrial process control problem', *Computers Chemical Engineering*, **17**, 245–255, (1993).

[7] D.B. Fogel, 'Ean introduction to simulated evolutionary optimization', *IEEE Transactions on Neural Networks*, **5**(1), 3–14, (1994).

[8] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, United States of America: John Wiley & Son, Inc., 1997.

[9] M.H. Hassoun, *Fundamentals of Artificial Neural Networks*, MIT Press, Cambridge, MA., 1995.

[10] K. J. Hornik, D. Stinchcombe, and H. White, 'Multilayer feedforward networks are universal approximators', *Neural Networks*, **2**(5), 359–366, (1989).

[11] J.C. Hoskins and D. M. Himmelblau, 'Artificial neural network models of knowledge representation in chemical engineering', *Computers & Chemical Engineering*, **12**(9), 881–890, (1988).

[12] T. Jockenhovel, L. T. Biegler, and A. Wachter, 'Dynamic optimization of the tennessee eastman process using the opt-control centre', in *Proceedings of the IEEE*, (2003).

[13] J. Kennedy and R. C. Eberhart, 'Particle swarm optimization', *Proceedings of IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ*, **5**(3), 1942–1948, (1995).

[14] J. Kennedy, R. C. Eberhart, and Y.Shi, *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco, 2001.

[15] R. Kiran, S. R. Jetti, and G. K. Venayagamoorthy, 'Online training of generalized neuron with particle swarm optimization', in *International Joint Conference on Neural Networks, IJCNN 06, Vancouver, BC, Canada*, pp. 5088– 5095. Institute of Electrical and Electronics Engineers, (2006).

[16] N. Kwok, D. Liu, and K. Tan, 'An empirical study on the setting of control coefficient in particle swarm optimization', in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, BC, Canada*, pp. 3165–3172, Vancouver, BC, Canada, (16-21 July 2006). IEEE Press.

[17] Ungar L.H., E. J. Hartman, J. D. Keeler, and G. M. Martin, 'Process modeling and control using neural networks', *AIChE Symposium Series*, **92**, 57–67, (1990).

[18] Jiann-Iiorng Lin and Ting-Yu Cheng, 'Dynamic clustering using support vector learning with particle swarm optimization', in *Proceedings of the 18th International Conference on Systems Engineering*, pp. 218–223, (2005).

[19] M.Norgaard, O.Ravn, Poulsen, and L.K.Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*, Springer, London, 2000.

[20] D. Montana and L. Davis, 'Training feedforward neural networks using genetic algorithms', in *Proceedings of Eleventh International Joint Conference on Artificial Intelligence*, pp. 762–767, (1989).

[21] T.J. Richer and T.M. Blackwell, 'When is a swarm necessary?', in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, eds., Gary G. Yen, Simon M. Lucas, Gary Fogel, Graham Kendall, Ralf Salomon, Byoung-Tak Zhang, Carlos A. Coello Coello, and Thomas Philip Runarsson, pp. 1469–1476, Vancouver, BC, Canada, (16-21 July 2006). IEEE Press.

[22] N. L. Ricker, 'Nonlinear model predictive control of the tennessee eastman challenge process', *Computers and Chemical Engineering*, **19**(9), 961–981, (1995).

[23] N. L. Ricker, 'Nonlinear modeling and state estimation of the tennessee eastman challenge process', *Computers and Chemical Engineering*, **19**(9), 983–1005, (1995).

[24] N. L. Ricker, 'Optimal steady state operation of the tennessee eastman challenge process', *Computers and Chemical Engineering*, **19**(9), 949–959, (1995).

[25] R.S. Sexton, R.E. Dorsey, and N.A. Sikander, 'Simultaneous optimization of neural network function and architecture algorithm', in *Decision Support Systems*, pp. 1034–1047. IEEE, (2002).

[26] A. Sheta and K. Eghneem, 'Training artificial neural networks using genetic algorithms to predict the price of the general index for amman stock exchange', in *Midwest Artificial Intelligence and Cognitive Science Conference, DePaul University, Chicago, IL, USA*, volume 92, pp. 7–13, (2007).

[27] A. Sheta, H. Turabieh, and P.Vasant, 'Hybrid optimization genetic algorithms (HOGA) with interactive evolution to solve constraint optimization problems', *International Journal of Computational Science*, **1**(4), 395–406, (2007).

[28] Alaa Sheta and H. Turabieh, 'A comparison between genetic algorithms and sequential quadratic programming in solving constrained optimization problems', *ICGST Internatioanl Journal on Artificial Intelligence and Machine Learning (AIML)*, **6**(1), 67–74, (2006).

[29] Alaa F. Sheta, *Modeling the Tennessee Eastman Chemical Reactor Using Fuzzy Logic*, volume 181, ISE Book Series on Fuzzy System Engineering-Theory and Practice, published by Nova Science, 2005.

[30] Chen Wah Sit, *Application of Artificial Neural Network-Genetic Algorithm In Inferential Estimation And Control of A Distilation Column*, Ph.D. dissertation, Universiti Teknologi Malaysia, 2005.

[31] John C. Sozio, *Intelligent Parameter Adaptation for Chemical Processes*, Ph.D. dissertation, Electrical Engineering Department, Virginia Polytechnic Institute, 1999.

[32] X. Yao, 'Global optimization by evolutionary algorithms', in *Proceedings of the 2nd AIZU International Symposium on Parallel Algorithms / Architecture Synthesis*, pp. 282–291, (1997).

[33] X. Yao. Evolving artificial neural networks, 1999.

[34] K. Zuo and W.T. Wu, 'Semi-realtime optimization and control of a fedbatch fermentation system', in *Computers & Chemical Engineering*, volume 24, pp. 1105–1109, (2000).

Real Time Movement Cooredination Technique Based on Flocking Behaviour for Multiple Mobile Robots System

Ghada AlHudhud[1] and **Aladdin Ayesh** [2]

Abstract. The emergent behaviour of a multiagent system depends on the component agents and how they interact. A critical part of interaction between agents is communication. This paper presents a multi-agent communication model for physical moving agents, robots. The scope of the paper emphasises the interaction relationships between individual components in multi-agents system that results in economical and social emergent behaviour. The proposed theory of communication integrates animal intelligence technique together with a cognitive intelligence one. This results in a local coordination of movements, and global task coordination. Accordingly, agents are designed to locally communicate with other agents in order to coordinate their movements via a set of flocking rules. A flocking algorithm is used because it satisfies a major objective, i.e. it has a real time response to local environmental changes and minimises the cost of path planning. A higher level communication mechanism is implemented for task distribution that is carried out via a blackboard-like conversation with a ground based controller.

1 Introduction

Recently, multi-agent systems have been considered an efficient tool for modelling the system intelligence and the corresponding emergent behaviour. This is because they possess common features; i.e. intelligence, autonomy, interaction through communication and co-operation. In addition, the recent advances in theories of agents and multi-agent systems (MASs) are currently contributing to diverse domains. An example of these domains is the multiple mobile robot systems ($MMRS$s) domain, which exemplifies the main features of the multi-agent systems.

$MMRS$ are in particular extensively studied for carrying out tasks that reduce human presence in certain or dangerous tasks, increase productivity, and reduce the time cost; e.g. surface planetary exploration, cleaning toxic wastes, or deep sea diving tasks. In these tasks, work can be performed more efficiently and reliably using several co-operative robots [6]. Modelling systems of multiple mobile robots can be carried out as a physical MAS model. Physical MAS models are particularly useful for systems that are composed of little robots moving in an environment and at the same time communicating with a higher layer agent; who can be considered as a ground based controller GBC. In this respect, agent's acts are classified into two categories: physical acts and communicative acts. The agent's physical acts requires movement co-ordination and the communicative acts require a central communication with controller.

A good co-operation requires sufficient communication. Therefore, it is essential to specify the level of communication required depending on the purpose of communication and the level of intelligence these communicating entities possess; e.g. movement action or information exchange action.

Specifying the level of intelligence a physical agent, robot, possesses is also important. Levels of intelligence could vary depending on whether it is the intelligence of the individuals that is required in the system or it is the system intelligence. Currently, these levels of intelligence are widely studied in the field of communication within Multi-Agent Systems (MASs) [5]. However, modelling a MAS requires that each individual agent must possess a high level of an interaction mechanism which when implemented generates global intelligent behaviour (good examples are presented in [11] and [12]).

Methods to generate global behaviours from many local behaviours have been applied to a varied sets of fields. An example is presented by Reynolds in [13] as a mathematical simulation of flocking behaviour. The flocking algorithm was first introduced by [13] as a technique that mimics the movements of bird flocks and related types of coordinated group motion. A set of flocking behavioural rules are described in details in [1], [2], [3] and [16]. Another example is presented by London [9] that introduced a new MAS model of price dynamics by considering some simplified cases to model the complexity and criticality in financial time series. Schlecht and Joseph also presented an example in [14], that implemented emergent behaviour techniques for modelling mission planning for unmanned air vehicles.

To sum up, the paper proposes a communication model that exploits the flocking algorithm as a movement co-ordination. The flocking algorithm is proposed to mimic the motion of a large group of animals. The action co-ordination concept exploits the blackboard communication technique, i.e. communication is performed by passing messages from a ground based controller GBC to agents for task allocation, and from agents to GBC in order to report status and findings. All tasks are distributed within a hierarchal order via a ground based controller.

2 Related Work

Despite the possible benefits of multiplicity, some problems are now introduced, for example path planning for a group of mobile robots. A cleaning task, presented in [10], and complex path planning presented in [8], requires either robots to have a pre-defined path plan to cover the unoccupied areas in a specified environment or an accurate map for the environment. For relatively small scaled systems, this has been shown to be feasible, whilst it is impractical to pre-specify the paths for a large number of agents or a dynamically changeable environment with moving agents.

An adaptive movement technique is required to co-ordinate move-

[1] Al-Ahliya Amman University, Jordan, email ghudhoud@ammanu.edu.jo
[2] De Montfort University,UK, email: aayesh@dmu.ac.uk

ments of a large group in a dynamic environment. An example of a system that uses an adaptable motion co-ordination technique is presented by Tang in [15], where an A^* path planning algorithm is used for searching an obstacle free path with the least cost for the agents from a starting point to the goal point but still expensive for the necessity to recalculate the navigation path. Considering the scenario where task assignment via a communication solution is to be added, methods, an example is presented in [4] and [7], addressed a lack of interactive task allocation algorithms and considered computationally intensive. The communication solution must measure the amount of communication required in order to control the agents' actions, whether these are movements or transmitting or receiving information. In addition, the way these agents communicate must be adaptable to the changes in the environment in which they are situated. Therefore, in order to overcome the above highlighted problems, a communication model is proposed that discards pre-specification and uses a mobile coordination and communication techniques and results in a swarming behaviour that exhibits suitable response actions while the agent is in motion.

3 New MAS Communication Model

Regarding the movement co-ordination, the flocking algorithm is used for local communication. This includes the identities of the sender and recipient, positions, and orientation. Hence, it considers the agents sensors as the only source of its knowledge. The flocking behaviour can be achieved by applying three rules: alignment, cohesion, and collision avoidance. Each rule, also considered as a subsystem, produces a suggestion for the interaction. These suggestions are simply: a centroid[3] agent's heading, a correction angle an agent needs to modify its direction, and finally a weighting value to determine the importance of this rule.

Three enhancements are proposed in order to address the list of problems with the conventional implementation of these rules. First, a local perception zone is used (that imitates the perception zone an animal may have in reality) rather than the conventional global one. This results in localising all computations so that only local agents are considered. The range of an agent's sensor controls the level of locality for an agent so that it can only interact with nearby agents. Therefore, the parameters for the perception zones are illustrated in table 1. The second enhancement is in the way an agent prioritises

Table 1: Perception zone for the three flocking rules

Interaction Rule	**Perception Zone**	
	S_d	FOV
Alignment Rule	$10\ units$	$360\ degrees$
Cohesion Rule	$10\ units$	$360 degrees$
Collision Avoidance Rule	$7\ units$	$36\ degrees$

its actions. Within the previous systems, the final decision is often made by giving the priority to the collision avoidance rule, i.e. if any object appears in the perception field, all the weights for the other rules will be reset until the agent avoids the object. According to the proposed new protocol, all the weights are considered regardless of the existence of any obstacle.

The third enhancement is the use of dynamically computed and distributed weights and rules. These weights are computed and dynamically updated each time an agent updates its sensory data. The relations between the weights and the centroids are empirically defined in tables **??** and 3. The conventional implementation of the flocking systems specifies homogeneous rules and weights in a way that all the agents use the same rules with the same weights. This results in global centroids and orientation.

[3] A centroid, in some cases, is an average spatial position for the positions of a set of agents. The centroid also can be the distance to a single agent, in other cases.

According to the above enhancements, the proposed new three flocking rules are described as follows:

- **Alignment Rule** R_α This rule is also known as a velocity matching rule as each agent A_i searches for the nearest neighbour from the same team A_j and tries to align its velocity vector with the velocity vector of this neighbour. The sensory data is filtered for this rule to pick only the nearest friend, a detected neighbour from the same team. The nearest neighbour is another agent that falls within a 360 degrees field of view and exists within a range equal to the agents sensor range. The sensor range and the field of view defines the perception zone for this rule.
  The alignment rule results in a velocity vector an agent should follow to modify its current direction in order to align with this neighbour, if found. For agent A_i, this vector composes a centroid $C_{A_i}^{R_\alpha}$ and a correction angle $\theta_{A_i}^{R_\alpha}$. The centroid of this rule is considered as the distance to agent A_j positioned at P_{A_j}, (see equation 1). The correction angle is computed as the difference between the current heading Ψ_{A_i} of agent A_i and the heading angle α_{A_j} of the nearest friend A_j, (see equation 2).

$$C_{A_i}^{R_\alpha} = P_{A_j} - P_{A_i} \tag{1}$$

$$\theta_{A_i}^{R_\alpha} = \Psi_{A_i} - \alpha_{A_j} \tag{2}$$

  This rule is implemented by the common flocking systems and this proposal adopts the same rule but with different weighting strategy. The weighting value $w_{A_i}^{R_\alpha}$ is dynamically updated and is computed as a reciprocal of the centroid magnitude. The weighting value is used to adjust the strength of the alignment force. The outputs from this rule $< C_{A_i}^{R_\alpha}, \theta_{A_i}^{R_\alpha}, w_{A_i}^{R_\alpha} >$ are stored in the internal blackboard and used to partially modify the agents current velocity.
- **Cohesion Rule** R_β
  The cohesion rule acts as a binding force. It reinforces each agent to orient its velocity vector towards the centroid of the team members that fall in the field of view of this rule (360 degrees) and exist within a range equal to the agents sensor range. For an agent A_i, located at corresponding positions P_{A_i}, the centroid $C_{A_i}^{R_\beta}$ of this rule is computed as the distance to the average of the positions of the m detected A_js located at position P_{A_j}s. The agent A_i computes the distance to the centroid $C_{A_i}^{R_\beta}$ and a correction angle θ^{R_β}.

$$C_{A_i}^{R_\beta} = \frac{1}{m}\sum_{j=1}^{m}(P_{A_i} - P_{A_i}) \tag{3}$$

$$\theta_{A_i}^{R_\beta} = \Psi_{A_i} - \beta_{A_j} \tag{4}$$

  Similar to the alignment rule, a weighting value $w_{A_i}^{R_\beta}$ is used to adjust the strength of the cohesive force. The weight value was empirically set according to the experiments described in section 4. The outputs from this rule $< C_{A_i}^{R_\beta}, \theta_{A_i}^{R_\beta}, w_{A_i}^{R_\beta} >$ are also stored in the internal blackboard.
- **Collision Avoidance Rule** R_γ
  This rule prevents an agent from colliding with other objects and agents in the environment. It also helps avoid overcrowding. An

agent uses the sensory data to specify whether it is safe to continue The perception zone of this rule differs from those of the alignment and cohesion rules. The range used for the collision avoidance is less than that used in both aligning and cohesion rules, the advantage of this being to avoid overcrowding only within a local area. In addition, an agents field of view for this rule is 36 degrees; starting from 18 degrees to the right and extending 18 degrees to the left of the agents velocity vector.
For an agent A_i, located at position P_{A_i}, the centroid $C_{A_i}^{R_\gamma}$ of this rule is computed as the vector to the detected object. Similar to the previously described rules, a correction angle $\theta_{A_i}^{R_\gamma}$ is set depending on the distance to the detected agent. Also, a weighting value $w_{A_i}^{R_\gamma}$ is used to give the priority to this rule over the other rules.

A higher level of communication between agents and human controller is performed through message exchange through a blackboard-like communication. All agents including the GBC-agent are able to read/write messages from/to this global blackboard. This is a significant part of the communication with the world, as the agent gains partial-global knowledge by communicating and negotiating with a GBC. This blackboard-like communication is implemented as a fourth rule in addition to the three flocking rules. Since the flocking rules are used to coordinate motion, the fourth rule is used to coordinate tasks and governs the global interaction with the GBC for task distribution.

Practically, this is done as follows: at the beginning of performing any specified task: the GBC issues a team task in the form of a target location for a set of agents. Each agent A_i in the team computes the expected distance $Dist_{A_i}^{tar}$ to a target (tar) positioned at (P_{tar}) according to,

$$Dist_{A_i}^{tar} = \|Pos_{A_i}^{start} - P_{tar}\| \tag{5}$$

where $\|Pos_{A_i}^{start} - P_{tar}\|$ is the Euclidean distance to the target. Next, an agent computes the number N of time intervals of length τ the agent needs to reach the target position depending on its speed Δx according to,

$$N = \frac{Dist_{A_i}^{tar}}{\Delta x} \tag{6}$$

Each agent has a counter C that counts the number of time intervals elapsed since the agent started the task and compares it to the expected N. Also, for each time interval τ , an agent estimates the remaining distance, with respect to its current position $Pos_{A_i}^{current}$, to the target which represents the centroid $C_{A_i}^{R_{BB}}$ of the blackboard rule.

$$C_{A_i}^{R_{BB}} = P_{tar} - Pos_{A_i}^{current} \tag{7}$$

Similar to the local interaction rules, a weighting value $w_{A_i}^{R_{BB}}$ is used to control the strength of this fourth rule. During all these stages the agent can regularly report its status via the external-blackboard to the GBC.

For each agent, and from each rule. the four rules produce four different suggestions an agent needs to consider when deciding the next action. Each suggestion is comprised of a centroid, a correction angle, and a weight. The information corresponding to each suggestion is stored in the internal blackboard. The strength of each suggestion is determined by the weights associated with each rule. These weights are computed and dynamically updated each time an agent updates its sensory data and /or recieves new task from the GBC. The relations between the weights and the centroids are empirically defined in tables **??** and 3.

Each suggestion represents a desired velocity demand. Hence, the new velocity vector is vector summed according to the following equations:

$$X_{A_i}^{\tau} = \sum_k w_{A_i}^{R_k} C_{A_i}^{R_k} \cos\left(\theta_{A_i}^{R_k}\right) \tag{8}$$

$$\Upsilon_{A_i}^{\tau} = \sum_k w_{A_i}^{R_k} C_{A_i}^{R_k} \sin\left(\theta_{A_i}^{R_k}\right) \tag{9}$$

From equation*5 ,6*, The new heading angle $\zeta_{A_i}$is calculated and is used to modify the agent's velocity and alter its motion in a suitable manner.:

$$\zeta_{A_i} = \arctan\left(\Upsilon_{A_i}^{\tau}, X_{A_i}^{\tau}\right) \tag{10}$$

4 Analysing Emergent Behaviour: Experimentation Results

This section tests, with the aid of visual evaluation. For this, a 3D simulation environment is being implemented in order to test how the interaction weights can be adjusted to help maximizing the coverage area by a group of agents, detecting the complexity in the environment depending on the information sent by a group of swarming agents.

4.1 Simulating the Communication Model

In order to simulate the communications between agents within a MAS model to mimic a real world, it is essential that the representation of these agents and their actions have a high level of realism. Simulating a MAS model, as a geometrical physical model implies that we have had to represent the environment and the agents with geometrical shapes. The agent's sensor is simulated as a fixed line segment at a defined angle centred at the agent's location but starting just outside of the agents physical space, see figure 1. The length of the line segment determines the laser sensor range. The sensor can rotate 360 degrees quantized into 18 $degree$ intervals and therefore it detects one of 20 different locations at each time interval (τ).

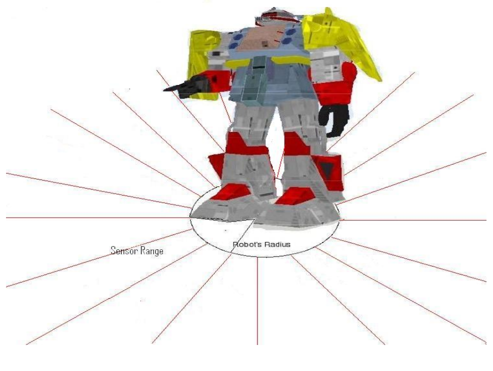


Figure 1: The simulated robot with laser sensor.

4.2 Controlling Interaction Weights

This section tests, with the aid of visual evaluation, how the user can adjust the interaction weights to help accelerating the agents' progress? For this purpose, controlling the interaction weights is carried out by running the Local Communication Model $(LC-Model)$, explained in section 3. As explained in section 3, the weights associated with the flocking rules are dynamically computed at each time interval τ depending on the rule's centroid and on the attitudes of each rule; these are the perception zone for this rule and the filtering strategy.

Table 2: Conventional Calculations of interaction weights

Alignment Rule	Condition	$\mathbf{w}^{\mathbf{R}_\alpha}_{\mathbf{A_i}}$
Centroide < seperation distance	$C^{R_\alpha}_{A_i} < S_d$	$1/C^{R_\alpha}_{A_i}$
Centroide > seperation distance	$C^{R_\alpha}_{A_i} > S_d$	1
Cohesion Rule	**Condition**	$w^{R_\beta}_{A_i}$
Centroide < seperation distance	$C^{R_\beta}_{A_i} < S_d$	$1/C^{R_\alpha}_{A_i}$
Centroide > seperation distance	$C^{R_\beta}_{A_i} > S_d$	1
Collision Avoidance Rule	**Condition**	$w^{R_\gamma}_{A_i}$
Centroide < seperation distance	$C^{R_\gamma}_{A_i} < S_d$	1
Centroide > seperation distance	$C^{R_\gamma}_{A_i} > S_d$	1

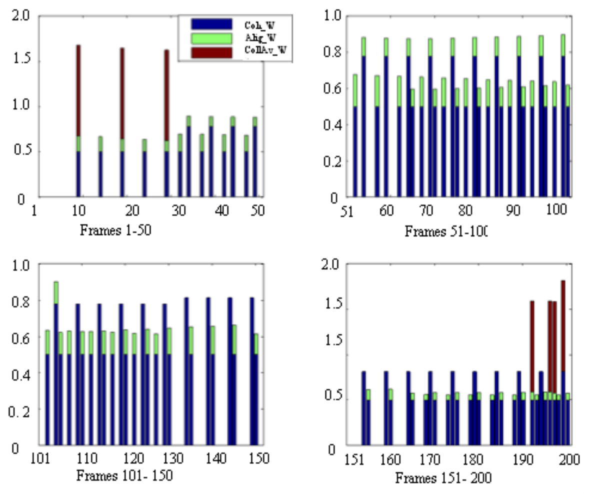


Figure 2: The interaction weights over the first 200 frames. The cohesion weight dominates the interaction weights whenever the avoidance weight is zero. The unmodified cohesion weight values are shown in table **??**.

Originally, the flocking system is implemented here with the weights computed in a conventional way, see table **??**. Since the collision avoidance weight is given precedence over other weights, as it is the most important interaction rule, the visual tests involved the influence of the cohesion weight ($w^{R_\beta}_{A_i}$) on the progress of the agents. The visual simulation has been useful at this stage in assessing and evaluating the extent to which varying the cohesion weight allows the agents in the same team to move as a unit. According to the visual evaluation, it was found that the cohesion weight slows the agents progress, due to the high influence of the cohesion force. In addition, it causes overcrowding in the surrounding area which is used as an indicator for examining the strength of this binding force.

Consider the situation where an agent detects a large number of nearby agents, then each of these agents modifies its velocity to move towards the cohesion centroid. If one or more of these agents detects a wall and at the same time some of the other agents within the avoidance zone, it may become trapped. In this trap situation, a neighbour of this agent (who may not detect the same objects) will be influenced by the trapped agent. In the same manner, the remaining agents will be influenced by the trapped agents as a result of a high cohesion weight. This can become worse if this set of agents is assigned a task to reach a specified target. Considering this scenario, the trapped agent continuously checks its capabilities of performing this task. Accordingly, the trapped agent may discard his commitment regarding completing the task. The other agents who detect the trapped agent will be influenced by the trapped agent which can still significantly slow their progress. This leads to a longer expected completion time, or even prevents the influenced agents from completing the task.

In this respect, a main goal of analysing the interaction weights then is to adjust the cohesion weight in order to avoid these impacts of a high cohesion weights without loosing the benefits of the supportive role of this weight in the team performance. Therefore, the start point was to test the conventional implementation of the weights in flocking algorithms, and the values are shown in table **??**, for the alignment $w^{R_\alpha}_{A_i}$ and cohesion weight $w^{R_\beta}_{A_i}$. For this implementation, the cohesion weight is computed as the inverse of the distance to the cohesion centroid ($C^{R_\beta}_{A_i}$) if the $C^{R_\beta}_{A_i}$ falls within the avoidance range, otherwise it set equal to one. This implies that the cohesion force is mostly inversely proportional to the distance to the centroid. The weight becomes bigger very quickly as the centroid position falls outside the avoidance range (S_d) whilst it does not become very small within the avoidance range. In order to numerically assess the dominance of the cohesion weight in situations where the agents do not detect any avoidance cases, the three flocking rules were used. Accordingly, these interaction weights are shown in figure 2. The bar graph shows the weights that control the strength of the interaction forces, according to the values shown in table **??**, on an agent over the first 200 frames of the simulation. Points of high cohesion weight, in figure 2, implies that an agent will be highly influenced by the nearby agents, and via monitoring the trap problem can be observed.

Table 3: The proposed interaction weights.

Alignment Rule	Condition	$\mathbf{w}^{\mathbf{R}_\alpha}_{\mathbf{A_i}}$
Centroide < seperation distance	$C^{R_\alpha}_{A_i} < S_d$	$1/C^{R_\alpha}_{A_i}$
Centroide > seperation distance	$C^{R_\alpha}_{A_i} > S_d$	1
Cohesion Rule	**Condition**	$\mathbf{w}^{\mathbf{R}_\beta}_{\mathbf{A_i}}$
Centroide < seperation distance	$C^{R_\beta}_{A_i} < S_d$	$1/(C^{R_\beta}_{A_i})^2$
Centroide > seperation distance	$C^{R_\beta}_{A_i} > S_d$	$1/C^{R_\beta}_{A_i}$
Collision Avoidance Rule	**Condition**	$\mathbf{w}^{\mathbf{R}_\gamma}_{\mathbf{A_i}}$
Centroide < seperation distance	$C^{R_\gamma}_{A_i} < S_d$	1
Centroide > seperation distance	$C^{R_\gamma}_{A_i} > S_d$	1
Collision Avoidance Rule	**Condition**	$\mathbf{w}^{\mathbf{R}_\gamma}_{\mathbf{A_i}}$
Carry on with the task	$N <$ expected time	1
Discard the atsk	$N >$ expected time	0

4.3 Coverage Areas

An attempt is made to explore the influence of the flocking emergent behaviour on the covered area around the target. These two issues

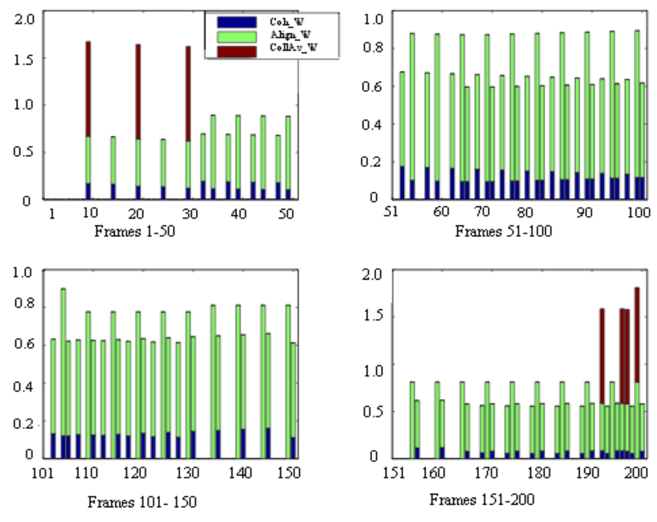
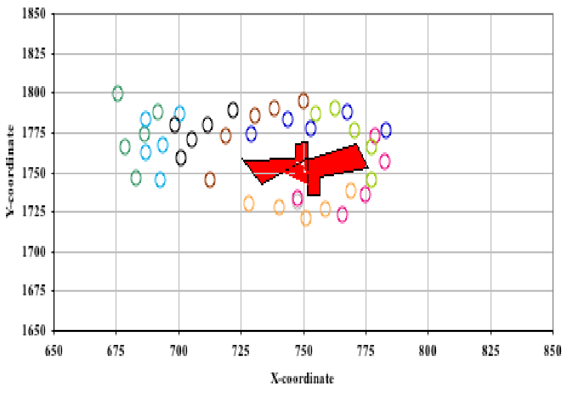
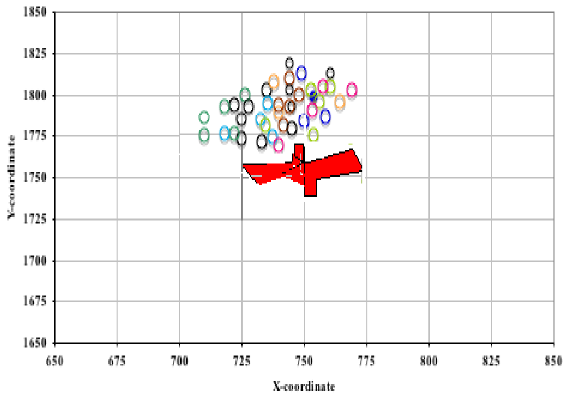


Figure 3: The interaction weights over the first 200 frames, with the cohesion weight modified according to the values shown in (table 3).



(a) ***Local-Global Communication***



(b) ***Global Communication***

Figure 4: The flocking behaviour, running the four communication rules, supports maximising the coverage area.

can be especially useful when the agents are to perform a search or sweeping task. This is carried out by viewing two values, the positions of the agents during movements and the area these agents cover after arriving at a specified target. Therefore, the experiment aims at running the model with a set of five agents forming one team, the four communication rules are switched on. After launching the model, these agents are issued a team command that informs them of the target location. On arrival, agents within a team will cover a wider area around the target position. This prevents the agents from overcrowding the target location and they are shown to appear to circle the target. Figure 4a, shows all the positions of the set of 5 agents running the local-global communication model over a period of time; note that agents swarmed about the location. The region of the covered area is computed as the number of occupied cells in the grid, each cell represents ($25\ cm \times 25\ cm$). This implies that as the number of occupied cells is 17, the agents cover $1.0625\ m^2$ during the last 6 frames. Comparing these results with those resulting from running the same number of agents communicate via the global communication rule only, figure 4b, the number of occupied cells is $9\ cells$ covering only $0.5625\ m^2$. This indicates that the coverage area by the flocking agents is about double that covered by individual agents.

Running the four communication rules implies that all the agents are committed to move within a team to reach the target location. On arrival, as the first agent moves forward the team centroid, which affects the cohesive force, also moves forward. Accordingly, the other agents who detect this agent also consider this team centroid in each calculation which implies they are pulling each other forward and at the same time towards the target location. These local interactions lead to the progress of the team centroid which in turn leads to the movement of the detected team members as rolling around the target location. Implementing the communication with the forth rule only, agents do not detect team mates in the neighbourhood hence agents intend to reach the target and only check for collisions. Therefore, these agents, on arrival, are either trying to avoid each other or looking towards the target which leads to a reduced possibility of covering a larger area.

4.4 Detecting Locations of Complexity within the Environment

A novel aspect of the numerical analysis of the interaction weights, was the ability by analysing the cohesion weights to extract ideas about the structure of the environment. This implies how the GBC can draw a rough idea about the locations of complexity by comparing the cohesion weights with the results of analysing teams x, y positions in terms of their means and standard deviations. Consider the communication with a higher level controller GBC, who could physically exist in a different location. The experiment is designed to run the simulation by launching a set of five agents in the simulated environment shown in figure 6 from a start point, at the front of the screen. Assigning a team task, to reach the target position shown in top left in the same figure. The numerical outcomes of the experiment consist of all the cohesion weights, and the (x, y) positions over a 500 frames.

The graph shown in figure 5, shows the distribution of the cohesion weight along the frames. The low values of cohesion weight indicates a large distance to the cohesion centroid. A small cohesion weight may represent those agents which are moving away from each other in an attempt to avoid an obstacle. This also may imply that the detected agents are not close to each other, in order to avoid one or

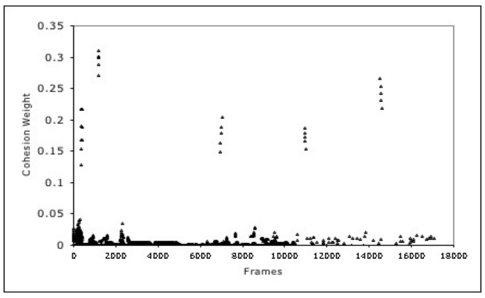


Figure 5: The distribution of the cohesion weights for each frame.

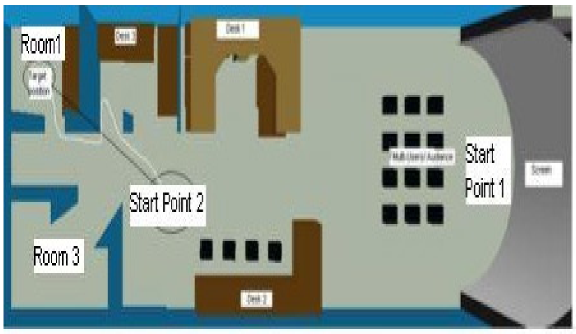
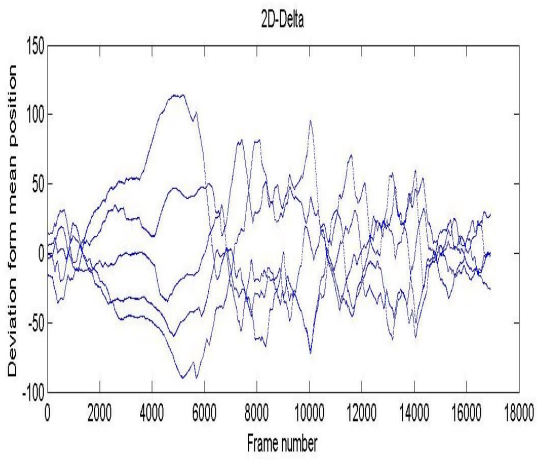


Figure 6: The Virtual Lab; simulated environment.

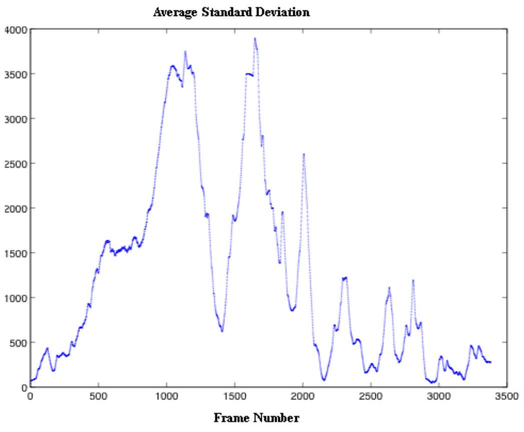
more obstacles, and may be splitting up, depending on their positions from the obstacle. The high values of the cohesion weights implies a reduced distance to the cohesion centroid, which indicates that the detected agents are close to each other. The graph in figure 5 can be considered as a way to extract locations of difficulties encountered in the environment. In addition, the test showed that the x, y positions can be fruitfully analysed together with the distribution of the cohesion weights to give a more explicit structure of the environment. Therefore, the x, y positions , sent by the agents to the GBC, are used to plot the deviations from the mean positions. During each frame, the mean positions of the agents and how far individually they are from the mean is calculated.

The graph in figure 7a shows the deviation of the agents' positions from the mean, for the task used to analyse these cohesion weights above. The deviation follows five different routes and shows the times of the obstacles encountered, as high deviations, and the times of open areas as low deviations. For example, the time encountered and locations of the four rows of chairs shown in figure 6, can be extracted. This also can be useful in assessing the agents ability to autonomously organise themselves by maintaining their positions with respect to their neighbours in order to act as a team on their way towards the target which supports the survivability aspect in the long term tasks. This level of autonomy is currently a highly demanded in robotic applications.

Since, the mean of the positions is sensitive to the change in the positions of the agents, the average distance from the mean location, standard deviation, is plotted in figure 7b. The standard deviation also indicates the locations of complexity as the high values show that these agents are not close to the mean position whilst the small values indicate the wide areas with no obstacles. Adding the information from both graphs 7b and 5 together shows that the low deviations in the former graph are represented as high weights in the later one, whilst the high deviations are represented as low weights in the graph. To conclude, the GBC can analyse the cohesion weights and can draw out an abstract picture of the environment by integrating the ideas from both graph 7b and graph 5, defining an explicit structure of the environment and the locations of complexity.



(a) ***The deviation of the agents positions from the mean***



(b) ***The average standard deviation for the same group***

Figure 7: The flocking behaviour, running the four communication rules.

5 Conclusion

The work presented in this paper describes a multi-agents communication model that serves well in controlling a multiple mobile co-operative robots system. The key form of co-operation is interaction. The communication model considers local movement co-ordination with nearby agents provided with animal-like intelligence. This technique provides economic computations for the movements of the multiple moving agents as agents only have a local view, local goals and knowledge. The global communication is carried out through blackboard-like procedure, where agents sends and receive information form and to the GBC via this blackboard. This prevents the disorder results when agents no longer possess a global view of the entire organization to which it belongs. A main advantage of setting the cooperation into this form is as follows, one hand agents cooperate to produce overall coordination and organization using a flocking behavior which resolves some of the conflicts that arises from the local interaction during the movement.

Implementing the flocking algorithm has been shown to be helpful when trying to control the motion of about $20 - 30$ moving objects which can be perceived as an organisation not a collection of individually moving agents. This is because it mimics the nature of a large organisation, examples are birds or herds of animals. However, in real life where a higher level agent, controller, tries to assign a task for this large group a concern is how to keep this controller in the loop.

ACKNOWLEDGEMENTS

Thanks to Dr. Martin Turner, University of Manchester/UK, for his support and co-operation; his comments helped improving the efficiency of the simulations performed. We would also thank Mr. Howell Istance, Head of De Montfort Virtual Environment Center/UK, for offering the chance to implement the system in the 3D environment.

REFERENCES

[1] G. Al-Hudhud, A. Ayesh, Martin Turner, and H. Istance, ‘Simulation and visualisation of a scalable real time multiple robot system’, in *Proceedings of the conference of Theory and Practice of Computer Graphics, TP.CG05*, University of Kent, Canterbury UK., (June 2005). Eurographics Association.

[2] Ghada Al-Hudhud, Aladdin Ayesh, Howell Istance, and Martin Turner, ‘Agents negotiation & communication within a real time cooperative multi-agent system’, in *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, pp. 611–617, Nottingham, United Kingdom, (December 16-18 2004). Nottingham Trent University. ISBN 1-84233-110-8.

[3] Ghada Al-Hudhud, Aladdin Ayesh, and Martin Turner, ‘Speech act and blackboard negotiation based communication protocol for real time multi-agent systems’, in *Proceedings of the UK Workshop on Computational Intelligence UKCI-2004*, pp. 112-120, Loughborough, United Kingdom, (September 6-8 2004). Loughborough University. ISBN 1-874152-11-X.

[4] R. Arthur, B. John, T. Micheal, and H. Jonathan, ‘Co-ordination and control of multiple UAVs’, in *AIAA Paper*, pp. 45-88. Guidance Navigation and Control Conference, (2002).

[5] J. Ferber, *Multi-Agent System and Distributed Artificial Intelligence*, Addison-Wesley, 2002.

[6] R. Grabowski, E. Serment, and P. Khosla, ‘An army of small robots’, *Scientific American, http://www.sciam.com*, (November 2003).

[7] T. Kam, T. Gregory, Z. Wayne, and T. Ann, ‘A multiagent operator interface for unmanned air vehicles’, in *Proceedings of the 18th Digital Avionics Systems Conference*, pp. 6.A.4.1–6.A.4.8, (October 1999).

[8] J. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[9] M. London, *Complexity and criticality in financial time series*, PhD. dissertation, De Montfort University, 2003.

[10] C. Luo, S. Yang, and D. Stacey, ‘Real time path planning with deadloock avoidance of multiple cleaning robots’, in *IEEE International conference on Robotics and Automation*, (September 2003). Tiwan,Thaibi.

[11] M. Mataric, ‘Designing emergent behaviours: from local interactions to collective intelligence’, in *From Animals to Animates 2, Proceedings of the Second International Conference on Simulation of Adaptive ‘Behaviour*, (1994).

[12] M. J. Mataric, ‘Learning to behave socially’, in *From Animals to Animates 3, Proceedings of the third International Conference on Simulation of Adaptive ‘Behaviour*, Brighton,D. Cliff, P. Husbands,J. -A. Meyer and S. W. Wilson (Ed), (1992).

[13] C. Reynolds, ‘Flocks, herds and schools: A distributed behavioral model’, in *SIGGRAPH ’87*, volume 21, pp. 25- 34, (July 1987).

[14] J. Schlecht. Mission planning for unmanned air vehicles using emergent behavior techniques, April 2001. Web Document available at http://www.cs.ndsu.nodak.edu/~joschlec/papers/uav_emergent.pdf.

[15] W. Tang, T. Wan, and S. Patel, ‘Real-time crowd movement on large scale terrains’, in *Theory and Practice of Computer Graphics*. IEEE Computer Society, (3-5 June 2003). ISBN 0-7695-1942-3.

[16] M.J. Turner, R. Richardson, A. Le Blanc, P. Kuchar, A. Ayesh, and G. Al Hudhud, ‘Roboviz a collaborative user and robot environment network testbed’, in *Interim Report for CompuSteer Workshop*. CompuSteer Workshop, (15th Sep 2006).

Aqua Swarms: Design and Implementation of Water Surface AUV

Mustafa Ozkan Daglioz and **Aladdin Ayesh** [1]

Abstract. Autonomous marine vehicles are remarkably useful for tasks such as shallow water surveying, environmental data collecting and military operations such as surveillance and weapon delivery. Because of the necessity for autonomous marine vehicles and the challenges they impose, they have gained a widespread interest among the research groups around the world. This paper presents a water surface autonomous unmanned vehicle which moves from a starting point to any desired destination while avoiding obstacles. Unlike many water surface unmanned vehicles this robots relies on oars for its mobility. This gives a greater control and precision in manoeuvrability while keeping the size of the robot as small as possible. This paper presents the hardware and software design and implementation of a dual-oar water surface autonomous vehicle. Comprehensive experimentation was conducted and results of the experiments are presented.

1 INTRODUCTION

The work presented here is to build a water surface autonomous unmanned vehicle (Aquabot) that uses two oars for its mobility. The main reason and a requirement is that the robot has great manoeuvrability and precise control. This Aquabot has been built using Lego Mindstorms Robotic Invention Kit. This robotic kit allows fast prototyping and flexibility in the robot structure. To present an alternative solution to traditional propelling methods used in autonomous marine vehicles, oars have been used to propel the Aquabot. The design presented here focuses on the performance, cost, environment, size and weight. The result is a blue print for a building block that can use to build and develop water swarms that are able to navigate in small spaces with restricted resources, i.e. light, maneuverability dimensions, etc.

The navigation and control system is designed to be a safe, effective and precise navigation system. To operate the Aquabot autonomously, software has been developed using Not Quite C (NQC) language, which is a language for programming a RCX microcontroller which control the movement of the Aquabot according to the signals received from light sensors fixed at the front of the robot (figure 1). The effect of the stimuli received through the light sensor lead to the deployment of one or both of the ora mechanisms.

The paper starts with a background section reviewing some of the Aqua and water surface robots developed and/or being researched. We then present the hardware design of our robot and show how it meets the requirement for small but efficient water surface unmanned vehicle. Software design follows. Details of experiments conducted and their results are then presented to conclude.

[1] De Montfort University, email: aayesh@dmu.ac.uk

2 BACKGROUND

Autonomous surface vehicles (ASV) are often used in marine tasks where the use of large maned ships is either impractical or dangerous such as the case in shallow water surveying, environmental data collecting, coordinating with autonomous underwater vehicles and military operations such as surveillance and weapon delivery. The variety of applications and the interesting technological challenges that building aqua robots imposes, led to an increased interest among research groups in developing marine robots often by multidiciplinary groups of marine researchers, mechatronics, and mobile robots.

ARTEMIS [1] was the first ASV developed by MIT. The vehicle was too small for open sea applications and it had some performance problems. In order to solve performance problems and increase the size, a catamaran hull was selected for ACES [1], which was the second ASV developed by MIT. Some changes were made to the hull, propulsion systems and power. After these modifications the ASV was renamed Autocad [2]. In 2004, four ASV named SCOUT [3] were developed by MIT. The vehicles consist of polyethylene kayaks equipped with lead acid batteries, communication systems and single board computer.

The MESSIN project [4], which was sponsored by the German Federal Ministry of Education Research and Technology, has been carried out from 1998 to 2000 to develop and test the ASV Measuring Dolphin for high accuracy positioning and measuring devices carrying in shallow water. In the period 1997-2000, the Instituto Superior Technico of Lisbon developed Delfim [5] [6] to develop an ASV for establishing a fast direct acoustic communication link between the AUV and a support vehicle. Lisbon IST is also developing Caravela [6] in addition to Delfim. Cavela is a long range autonomous research vessel, for testing advanced concepts in vehicle/mission control and radar based obstacle avoidance and demonstrating technologies for the marine science community [7].

Charlie, an autonomous catamaran, developed in 2002-2004 by CNR-ISSIA. Charlie initially designed for supporting sensors and samplers for the study of the sea-air interface in Antarctica, where it was exploited in 2004 [8]. In 2005, original steering system was upgraded and the vehicle is currently used for testing of mission control, navigation and guidance algorithms and for evaluating the possibility of using this technology for civil harbour protection in the presence of marine traffic [7]. University of Plymouth, UK, is developing an ASV named Springer for tracing pollutants. The unmanned Springer will be around 3m long and 1.5m high [9].

In all of these projects, issues of hardware and software design were identified often leading to a revision in the design, e.g. [1]. The dynamics of water surface, shape of the robot, and the emerging responses of the interaction of the different forces make hardware design an important issue. The hardware design impact on the software, which has to respond to the sensors in use. The software and hardware of the Aquabot proposed here have been designed considering the problems highlighted by the reviewed projects and in response to requirements of this project.

The main hardware problems that ASVs have can roughly be gathered under three title; buoyancy, size and weight. Projects mentioned here have different hull shapes and propulsion methods in order to solve these three problems. We studied the drawbacks of previous designs and reflected proposed solution in our design. Many ASVs reviewed here have a very complicated software to control and operate the hardware. Due to the limited capacity of the software platform of the Aquabot, complicated navigation methods of the projects were simplified using swarm technology and thus a simple but efficient reactive controller was implemented.

3 HARDWARE DESIGN

3.1 Design Considerations

Hardware design is probably the most important part of designing an Aquabot. Without proper hardware design, the Aquabot cannot float on the water or perform necessary actions to reach the destination point. The Aquabot has been designed to meet these 5 requirements; cost, performance, environment, size and weight. The design explained here gives the Aquabot precise and efficient propulsion ability. Since the Aquabot has been designed considering overall weight and size, it does not need much power to propel itself. This is also important because RCX microcontroller, which is the power source of Aquabot, uses 6 AA size batteries as a power source.

Another important consideration is buoyancy. With this design, Aquabot has an equal distribution of weight, which provides well balanced buoyancy. At the end, a small, lightweight and yet efficient Aquabot has been designed.

The working of the Aquabot can be summarised as follows; when Aquabot is running, to detect the location of the destination point, it waits for the signal from the destination point. Aquabot has been adjusted to measure the distance between the source of the signal and itself. The best way of sending signals from the destination point is to use a laser pointer. Therefore Aquabot measures the intensity of the laser that has been sent from the destination point and using that information Aquabot calculates the distance between the destination point and itself. After detecting the location of the destination point, Aquabot starts moving towards it. Software has been designed to keep the Aquabot away from obstacles. When Aquabot hits an obstacle, it moves around the obstacle and keeps going to the destination point.

3.2 Mechanical Structure

The Mechanical structure of Aquabot consists of 5 parts. These are; the hull, oars, oar mechanism, obstacle sensing mechanism and microcontroller. These 5 parts have been designed individually, then fixed together to form the Aquabot. Every one of these 5 parts has its own design considerations. These considerations and detailed mechanical design of each part will be explained individually in this section. Design of the hull will be explained first, then the other parts will be explained.

The hull is the body of the Aquabot, and selecting an appropriate hull form is important in terms of stability, capacity of payload transport and floatability. A catamaran design has been selected as a hull form in this piece of work. This is because waterplane area between two hulls reduces rolling motions and increases displacement. Since catamarans consists of two hulls, failure of one hull does not end up in a complete loss of buoyancy. To build a catamaran hull, two model ship hulls have been used. Two hulls fixed to each other with 2 cm. space between them. The hull can be seen in the figure 1.

3.3 Oar Mechanism

The oar is the part of the Aquabot that steers, accelerates and stops the boat. Four factors have been taken into consideration while designing oars. These are weight, size, strength and stiffness. To meet these factors, plastic material has been used to build oars. In terms of making more efficient oars, the blades of oars have been curved. With this design, 94 percent of the oar surface touches water. Therefore the oars transfer enough power from motors to the water to propel Aquabot. The Oars can be seen in figure 1.

The oar mechanism is the mechanism that moves the oar in vertical and horizontal directions. In other words it is the main mechanism that drives the Aquabot. It provides power and transfers the power to the oars. Aquabot has one oar mechanism for each oar. Mainly, the oar mechanism consists of two motors, two gears, two bars, one oarlock and one touch sensor. To move the oars horizontally, a slide mechanism has been used. The body of oar mechanism has been attached to a ring, and the ring has been attached to two bars. Both ends of the bars have been fixed to the hull. Therefore the body of oar mechanism can slide in a horizontal direction. One motor and one gear have been used to slide the body of the mechanism. To move oars vertically, the gear mechanism has been used. A Long black beam has been used to connect the oar to the gear. The idea behind the gear mechanism is this; every half cycle of the gear, the black beam will move up or down, therefore the oar will move up or down. To limit the movement in vertical direction, 1 Degree Of Freedom (DOF) joint has been used. To stop the motor at the end of every half cycle, a touch sensor has been used. To propel Aquabot, vertical and horizontal motions of the oar mechanism are performed in order. This mechanism can be seen in figure 1.

Each oar mechanism has been placed on each hull. To propel Aquabot straight forward, two oars must move simultaneously. To achieve this aim, two oar mechanisms have been connected to each other using two beams. The obstacle sensing mechanism has been placed in front of the Aquabot. The RCX microcontroller has been fixed to the middle of the hull in terms of balance. One light sensor has been fixed on top of the RCX microcontroller. This light sensor has been used for navigation. Connection cables between the microcontroller and the motors or sensors have been fixed to the hull. The oars have been fixed to the hull using the oarlock, which lets the oar rotate around. Dimensions of the Aquabot; Width = 21 cm.(without oars), Length = 38 cm, Depth = 11 cm. (4,5 cm of 11 cm. is under water.)

3.4 Sensing and Control

The obstacle sensing mechanism is a specially designed mechanism that is used to sense the obstacles in front of the Aquabot. The obstacle sensing mechanism consists of 3 main parts. These are; touch sensor, spring and bar. Two bars have been connected to the front bumper. The bars have been fixed to the hull with help of two springs. The touch sensor has been placed near the other end of the bars. Every time Aquabot hits an obstacle, the front bumper pushes the two bars, the two bars push the touch sensor and with the help of springs the two bars and the front bumper go back to its former position. With help of this mechanism, every time Aquabot hits an obstacle, the touch sensor sends a signal to the microcontroller. This mechanism can be seen in figure 1.

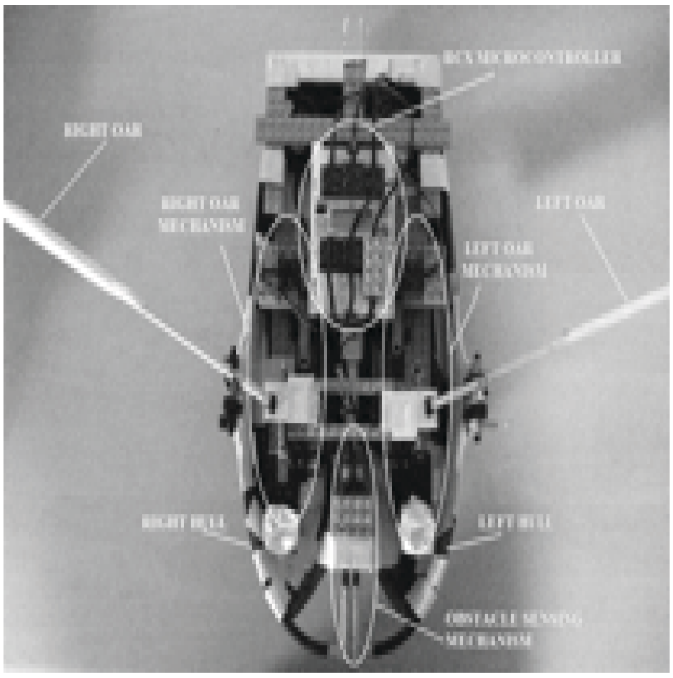


Figure 1. Aquabot: dual-oars water surface swarm robot

RCX microcontroller is the essential part of the Aquabot. RCX microcontroller contains three sensor inputs, three actuator outputs, an IR transceiver, Serial I/O (input/output), ADC (analog digital converter), 8 bit CPU at 16 MHz and Hitachi H8 microcontroller with 32 kilobytes of RAM (4 kilobytes of which is used for interrupt vectors and other low level data.) User programs are stored in 16kb of internal ROM and 32kb of static RAM. An ADC (Analog to Digital Converter) lets the RCX read sensor inputs, and special driver chips makes controlling motors and electrical devices possible. As an user interface, microcontroller has four user buttons, LCD display and an internal speaker. Microcontroller is the heaviest part of the Aquabot. Because of this reason, it was essential to find an appropriate place to fix it. After some calculations, microcontroller has been fixed to the 3 cm away from the middle point of the Aquabot. Therefore, microcontroller does not relocate the centre of gravity of the Aquabot and Aquabot does not lose its balance.

4 SOFTWARE DESIGN

Navigation has always been a difficult task in robotics. Autonomous navigation means that a vehicle can move to a desired destination without a user interference. To design a good autonomous navigation system, every bit of information should be collected from all available sources. In this piece of work, these sources are the sensors. At this point, there is a limitation on the number and types of sensors that can be used with the RCX microcontroller. There are only three sensor inputs on the RCX microprocessor. Two sensor inputs have been used to control the oar mechanisms. Therefore only one sensor has been used to navigate the Aquabot.

The software has been programmed using Not Quite C (NQC). A NQC program is composed of code blocks and global variables. There are three distinct types of code blocks: tasks, inline functions, and subroutines. Each type of code block has its own unique features and restrictions, but they all share a common structure. Tasks can be run simultaneously. Because of this advantage, most of the program has been composed of tasks. The main task is started whenever the program is run. Other tasks can be started from inside the main task. Two tasks have been written to control the vertical motion of each oar. Since the two oars have been connected to each other, one task has been written to control the horizontal motion of the two oars. Propulsion of the Aquabot is done by running these tasks continuously. Another task has been used to detect obstacles. This task is run simultaneously with the main task when the program starts. As it was mentioned earlier, the main task waits for the signal from the destination point to detect the location of said destination point. After this signal, Aquabot starts moving. If Aquabot does not receive a signal for 60 seconds, it turns around to wait for a signal from another direction.

Navigation system approaches can roughly be divided into two groups, absolute positioning and relative positioning [10]. Absolute positioning is when the robot uses landmarks to determine its position. Since landmarks can not be placed on water, the only solution for the Aquabot is to use relative positioning. Relative positioning does not require landmarks. In relative positioning, Aquabot computes its position from the staring point. This is done by counting strokes that Aquabot takes. To count strokes, the program assigns a variable which equals zero at the beginning. Every time Aquabot takes a stroke, the program adds one to the variable, if Aquabot goes back because of an obstacle, the program subtracts one. With one stroke, Aquabot moves 45 cm away. Using this information, the program converts the distance to number of strokes. Therefore the program knows how many strokes that Aquabot has to take in order to reach the point of destination. When the stroke counter variable equals the number of strokes that Aquabot has to take, Aquabot knows that it is at the destination point and stops.

To avoid obstacles, the program does the following; four variables have been used for four directions. These directions are; North, South, East and West. The program assumes that Aquabot always starts to move northward. When Aquabot starts moving, the program assigns a number of strokes that it has to take in order to reach the point of destination to the north variable. Therefore the program knows that Aquabot has to go to the north to reach the destination point. Aquabot starts going north until it reaches the destination or hits an obstacle. When Aquabot hits an obstacle, the program assigns two more strokes to every variable of every direction. Therefore, Aquabot knows that it has to go two strokes south, then two strokes east, then two strokes north and then two strokes to the west to avoid obstacles. With the help of this algorithm, Aquabot moves around the obstacle. After avoiding the obstacle, Aquabot keeps going to the point of destination. To monitor which direction Aquabot is going in, another variable has been used. Every time Aquabot turns right the program adds one to the variable, or subtracts one if it turns left. Therefore the program always monitors to which direction Aquabot moves. The program repeats the same set of sequences every time Aquabot hits an obstacle. Therefore, Aquabot can avoid very complicated obstacles.

The flowchart of the navigation controller can be found in figure 2. To keep flowchart simple, each task is represented by a square. Therefore simultaneous run of the tasks can be seen clearly.

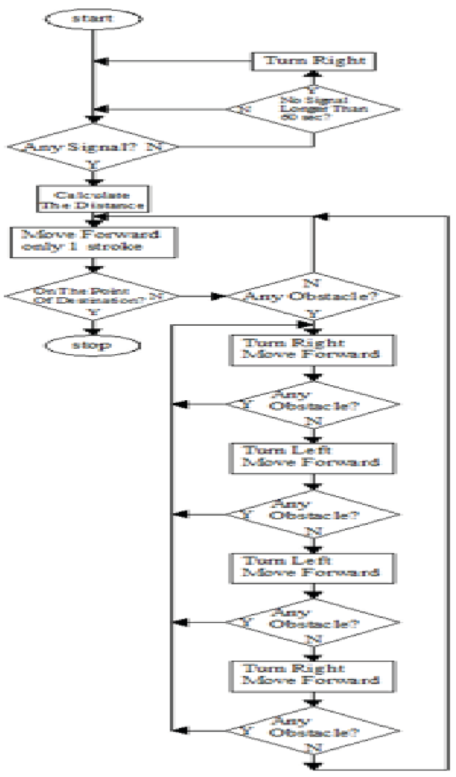


Figure 2. Flowchart of the navigation controller

5 EXPERIMENTS

5.1 Experiment Settings

The first group of experiments has been done to test the obstacle avoidance system. To test this system, four different obstacles have been used with different start and destination points. Schematic representations of these experiments can be found in figure 3. In the first experiment a small simple obstacle has been used. As it can be seen in figure 3, Aquabot followed the route that has been indicated with black line. When Aquabot hit the obstacle, it went back then moved around the obstacle and reached the destination point. In terms of avoiding obstacle, the first experiment was successful. In the second experiment, more complicated obstacle have been used. Because of the size of the obstacle, Aquabot hit the obstacle three times, but it managed to manoeuvre around the obstacle and reach its destination point. A bigger obstacle has been used in the third experiment. As can be seen on the figure 3, Aquabot hit the obstacle four times. But in the end, the experiment was successful; Aquabot reached its destination point. The fourth experiment has been done to find the answer of the question of what happens if the destination point has been surrounded by obstacles. As can be seen in figure 3, this experiment was also successful, the program managed to lead Aquabot to its destination even though the destination point has been surrounded.

The second group of experiments have been done to determine the range of the signal receiving system. Experiments show that the program successfully receives signals in the range of 50 cm to 7 m. As has been mentioned in previous sections, the light sensor has been used to receive signals. The light sensor measures the intensity of light and according to intensity, it sends a value to the RCX microcontroller. Even if there is no signal from destination point, the light sensor measures the intensity of daylight. Because of this, the RCX microcontroller cannot receive signals from a distance greater than 1,5 m in very bright daylight. In average daylight it can receive signals up to 3 m. And it works best in dark environments, it can receive signals from 7 m.

5.2 Experiment Results

Aquabot has shown good performance in most of the experiments. Aquabot has successfully received signals from the destination point, accurately measured the distance between the destination point and itself, propelled itself with oars, avoided obstacles and successfully reached the destination point. These experiments show that the aims mentioned previous sections were accomplished.

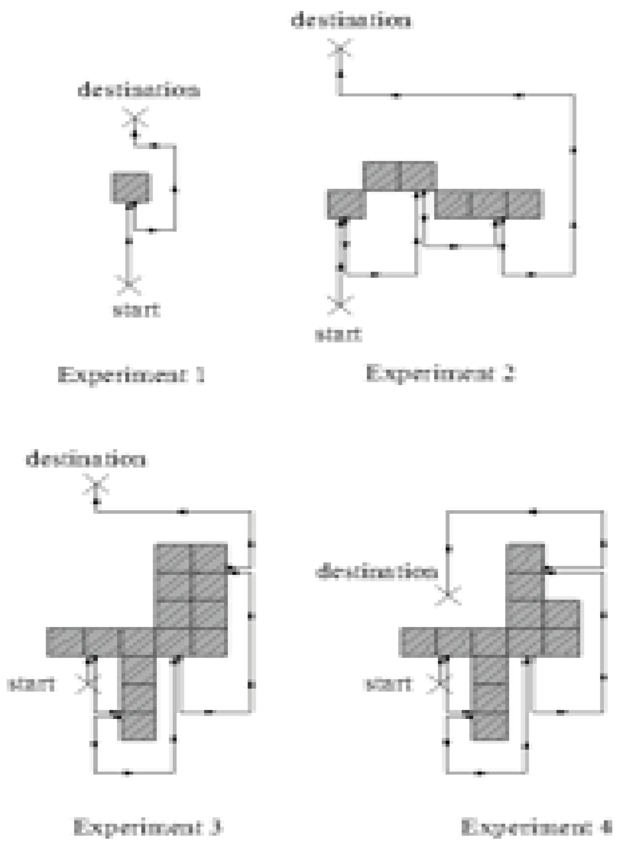


Figure 3. Experiment Settings

The summary of the results of the four experiments that can be seen in figure 3, can be seen in table 1. These results show that the length of the route followed by Aquabot changes depending on the area of the obstacle. In addition to the area of the obstacle and the length of the course, the number of hits occurred during experiments and the distance between start and destination point are also mentioned in the said table.

Two pictures taken during experiments can be seen in figure 4 and 5. In figure 4, Aquabot is approaching the obstacle. Figure 5 was taken seconds after the first picture while Aquabot avoiding obstacle.

The full length video of one of the experiments is uploaded on

Table 1. Summary of the Experiments

No. of Exp.	◇	□	△	⋆
1	0.5 m	0.92 m	81 cm^2	1
2	0.82 m	3.03 m	486 cm^2	3
3	0.64 m	2.74 m	1134 cm^2	4
4	0.28 m	3.10 m	972 cm^2	4

◇ Distance □ Course Length △ Obstacle Area ⋆ Hits Occurred

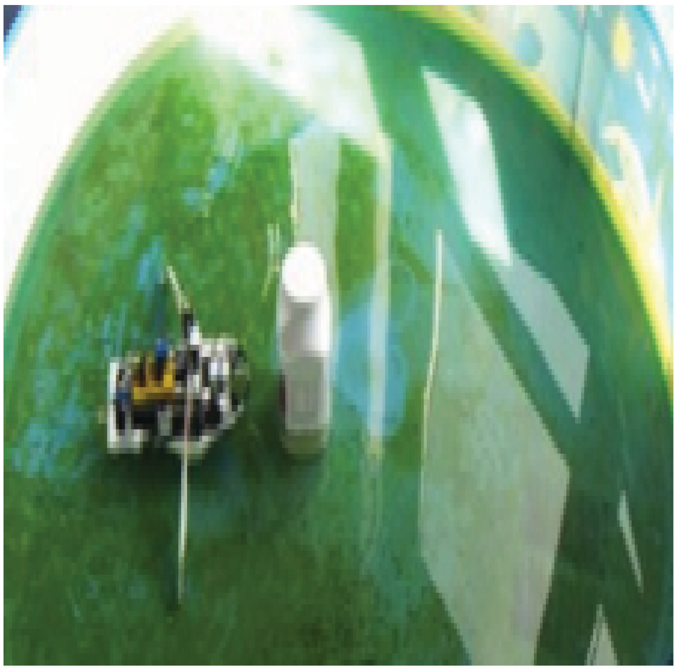


Figure 4. Still Picture from Experiment

youtube [2]. In this video, behaviour of the Aquabot during obstacle avoiding experiment can be observed. An obstacle smaller than the obstacles used in four experiments has been used in the video. Despite of small obstacle, this video demonstrates a good example in terms of navigation and obstacle avoidance.

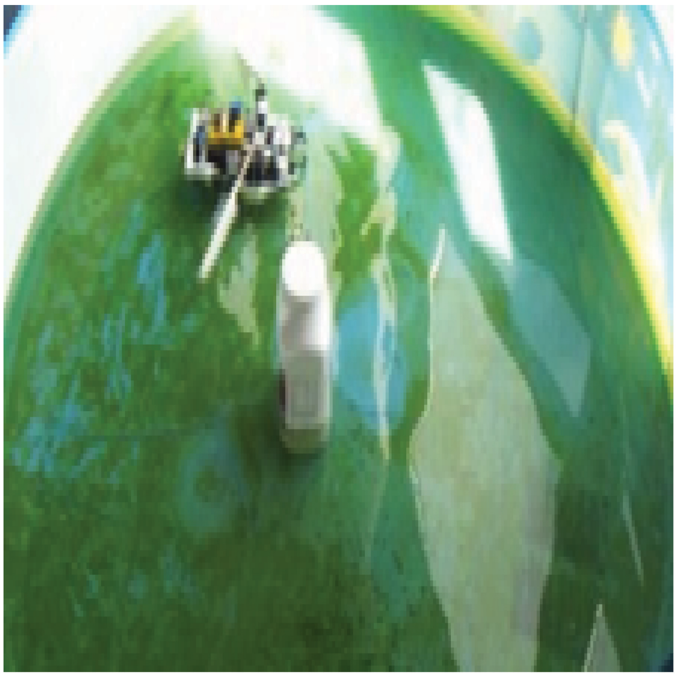


Figure 5. Another Picture from Experiment

6 CONCLUSION

The above experiments have proven the reliability and successful working of the Aquabot. The first lesson that can be learned from the work described above is that with Lego Mindstorm Robotic Invention Kit, efficiently working robots can be built. It was easy to build hardware part of the Aquabot with Lego parts, but on the other hand this easiness brought some limitations with it. For example, because of the location of the obstacle sensing mechanism, the front part of the hull of Aquabot hits the obstacle. Otherwise Aquabot can not detect obstacles. In experiments it was observed that when one of the oars of the Aquabot hit the obstacle, Aquabot loses its balance. This limitation can be eliminated using another design for the obstacle avoidance mechanism.

Another limitation of the Aquabot happened because of the Lego light sensor. The range of the Aquabot depends on the daylight. Since the components that we can use are limited with Lego parts, it is very hard to eliminate this limitation. A more sensitive light sensor could be used in the Aquabot, but there is not another light sensor in the Lego Mindstorm kit.

Another limitation happened because of the lack of the relative positioning. In relative positioning, Aquabot computes its position from the staring point. The program assumes that if Aquabot takes one stroke, it goes 45 cm away from the start point. In experiments it was seen that depending on the flow rate and direction of the flow,

[2] The link of the video is; http://www.youtube.com/watch?v=1PJGxYXFMsM

this length was changed. Experiments also show that if something disturbs Aquabot and makes it move without control of the program, Aquabot can not find the destination point. This problem could be solved using a PID controller. To implement this controller, more than three sensors must be used, which is quite impossible because the RCX microcontroller only has three sensor inputs and all of them were used to control the oars or to receive signals.

All limitations mentioned in this paper occurred because of the available hardware. But even with these limitations, experiments showed that the aims of the work were accomplished.

With help of IR transceiver and sensors, Aquabot can easily communicate with other robots around. This communication could help the Aquabot to gather any necessary information required to successfully accomplish missions. This advantage of the Aquabot is very important in terms of extended swarm robotics employing direct communication mechanisms as part of their swarm algorithm. The water surface AUV implemented here has been designed to operate as an autonomous entity individually or in collective producing a complete random swarming behavior. However, the same technical specifications provided in this paper, with adjustment to software implementation, can enable this AUV to work with other robots in a goal-directed fashion.

7 REFERENCES

[1] Manley, J. (1997) Development of the autonomous surface craft ACES. Proc. of Oceans97, vol. 2, pp. 827 - 832.
[2] Manley, J., Marsh, A., Cornforth, W., Wiseman, C. (2000) Evolution of the autonomous surface craft AutoCat. in Proc. of Oceans00, vol.1, pp. 403 - 408.
[3] Curcio, J., Leonard, J., Patrikalakis, A. SCOUT - A low cost autonomous surface platform for research in cooperative autonomy.
[4] Majohr, J., Buch, T. (2006) Advances in unmanned marine vehicles. IEE Control Series, ch. Modelling, simulation and control of an autonomous surface marine vehicle for surveying applications Measuring Dolphin MESSIN, pp. 329 - 352.
[5] Pascoal, A., Silvestre, C., Oliveira, P. (2006)Advances in unmanned marine vehicles. IEE Control Series, ch. Vehicle and mission control of single and multiple autonomous marine robots, pp. 353 - 386.
[6] Pascoal, A. (2000) Robotic ocean vehicles for marine science applications: the european asimov project. in Proc. of Oceans 2000.
[7] Caccia, M. (2006) Autonomous Surface Craft: Prototypes And Basic Research Issues. Control and Automation. MED ’06. 14th Mediterranean Conference. pp. 1 – 6.
[8] Caccia, M., Bono, R., Bruzzone, G., Spirandelli, E., Veruggio, G., Stortini, A., Capodaglio, G., (2005) Sampling sea surface with SESAMO. IEEE Robotics and Automation Magazine. vol. 12. no. 3. pp. 95 - 105.
[9] www.plymouth.ac.uk/pages/view.asp?page=9007.
[10] Horn, M. (2005) Developing safety-security critical systems: A prototype LEGO Mindstorm Detection System. Norwegian University of Science and Technology, Software Engineering Depth Study.
[11] Breivik, M., Fossen, T.I. (2004) Path Following For Marine Surface Vessels. Mts/Ieee Techno-Ocean ’04, 4. pp. 2282 - 2289.
[12] Encarnacao, P., Pascoal, A. (2001) Combined Trajectory Tracking And Path Following: An Application To The Coordinated Control Of Autonomous Marine Craft. Decision and Control. Proceedings of the 40th IEEE Conference, 1. pp. 964 - 969.
[13] Naeem, W., Xu, T., Sutton, R., Chudley, J. (2006) Design Of An Unmanned Catamaran With Pollutant Tracking And Surveying Capabilities. UKACC Control. Mini Symposia. pp. 99 – 113.
[14] Martin, F. G. (1995) The Art Of Lego Design. The Robotics Practitioner: The Journal for Robot Builders, 1 (2).
[15] K.A.Hawick, K., A., James, H., A. (2002) Simulating Swarm Behaviour of Robots. Technical Note.DHPC-118, Submitted to IASTED Conference on Applied Modelling and Simulation.
[16] Mondada, F., Pettinaro, G., C., Guignard, A. (2004) Swarm-Bot: A New Distributed Robotic Concept. Autonomous Robots. Vol. 17. No. 2-3. pp. 193 – 221.

Ant Colony Optimisation for Large-Scale Water Distribution Network Optimisation

Laura Baker[1], **Ed Keedwell**[1], **Mark Randall-Smith**[2]

Abstract. In this paper we show that ant colony optimisation (ACO) can be successfully applied to large-scale water distribution network optimisation problems. In addition, a new ACO algorithm ERMMAS is proposed and is tested on this problem. A water distribution network taken from industry is optimised by a number of ACO systems and a well-tuned GA. The results indicate that although there are not large-scale efficiency savings to be made, ACO is capable of finding results of equal or better optimality than a comparable GA.

1 INTRODUCTION

Water distribution networks (WDN) serve to transport clean water from treatment works to individual customers and usually represent a significant capital investment in the development of the urban environment. The problem of designing a WDN to optimally meet performance criteria, such as delivering sufficient water pressure for high rise buildings and fire fighting; whilst minimising cost criteria, such as the cost of material, excavation, frequency of maintenance is known to be NP hard. A large variety of computational algorithms have been devised for this task which include well known techniques in operational research such as linear, dynamic and integer programming. In recent years however, a variety of nature-inspired and meta-heuristic algorithms such as genetic algorithms, simulated annealing and tabu search have been widely investigated as useful research tools for WDN design. Amongst these meta-heuristic algorithms, genetic algorithms (GAs) has proved to be one of the most popular with the application of GAs to WDN optimisation tracing back to the mid-nineties (Dandy et al., 1996; Savic and Walters, 1997). Whilst GAs have provided good solutions to water distribution optimisation problems for some time, the steady increase in the complexity of the network information being kept by the water companies means that GAs are no longer always suitable. This is in part due to the long running times incurred by the algorithm due and in particular, the high number of objective function evaluations required by evolutionary techniques. An increasing number of elements in the network and more detailed 24-hour simulation studies has seen the complexity of a single network simulation increase massively. Therefore, researchers are constantly looking for techniques which might deliver GA-class results, but with fewer objective function calculations. In this paper we investigate the application of a swarm intelligent approach to the problem of water distribution network optimisation. We describe the application of three ant colony systems to a large-scale water distribution network taken from industry and compare it with a well-tuned GA, with mixed results. The remainder of this section discusses water distribution network optimisation and previous research into using ant colonies for this purpose.

Water Distribution Networks

WDNs are part of the water supply system, comprising of number of interconnected elements such as pipes, nodes, pumps, valves, and reservoirs of varying shapes and sizes. The nodes represent combined points of water demand (e.g. housing or industrial estates) on the system. The purpose of the network is to deliver water to the demand nodes from the water treatment works, reservoir, or other source throughout the day and under varying demand conditions. The demands on a WDN fluctuate throughout the day. Peak demands occur when people prepare to leave for work at around 7am until 9am and when industrial organisations begin work for the day. It is important that demands on the network at peak times are satisfied. However, WDNs are costly to construct, maintain and operate hence the need for the optimal design of WDNs, where least cost can be balanced with required water pressure levels.

There are many options to be considered when optimising a WDN, but in most case, an existing network is already in place making it difficult to attempt major structural change in the existing design. Changing the position of the network elements is considered a major structural change and would be very costly and therefore many studies are restricted to the rehabilitation of components within the network. This is achieved by replacing existing components which no longer meet the demands placed on the system with more suitable infrastructure. However, as this is a large capital investment, the water companies inevitably want these modifications to last for long time periods, typically 50-100 years. Therefore rehabilitation studies investigate changes such as replacing pipes, pumps and tanks with different sizes/specification of the same element can have a large effect on the performance of the network.

[1] School of Engineering, Computing and Mathematics, University of Exeter, Harrison Building, North Park Road, Exeter, UK, EX4 4QF Email: {lb281, E.C.Keedwell}@ex.ac.uk

[2] Mouchel, Priory Court, Poulton, Cirencester, Glocs, UK GL7 5JB Email: mark.randall-smith@mouchel.com.

Attempting to find the optimum design for a WDN by hand is an arduous task, but engineers have done this in the past. AI techniques (e.g. the GA) make developing a proposed solution much quicker and easier although it can be difficult to align with engineering expectations. To perform an exhaustive search is implausible even with a small network which is demonstrated by a simple example known as the New York Tunnels network (Schaake and Lai, 1969) shown in Figure 1.

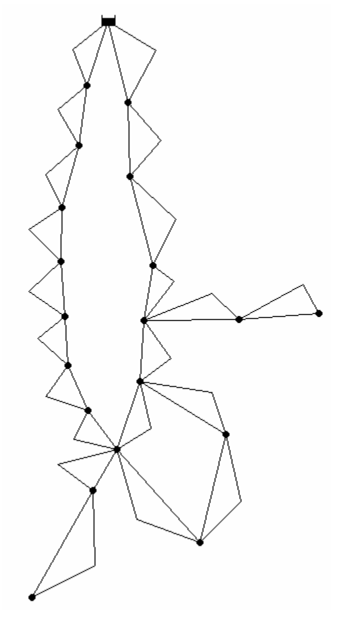


Figure 1 - Schematic of the New York Tunnels network. A single reservoir at the top feeds demand nodes located around New York via a set of duplicated pipes.

The optimisation algorithm must determine the pipe diameters of the 'new' pipes which represent expansions to the existing system. The algorithm has 16 possible commercial diameters to select from and there are 21 pipes to size, each with a potential effect on the other sizes in the network. Therefore, even in this small network there are 16^{21} or $1.93*10^{25}$ possible combinations of diameters to select from. This gives an indication of the problem complexity faced by the optimisation algorithm even for small problems.

The Example Network

The water distribution network used in this study is a real network taken from industry and represents the entire network for a large North American city. The attributes of the network taken from a hydraulic simulator are shown below:

Table 1 - Element counts of the example network

Element	Number
Number of Junctions	543
Number of Reservoirs	4
Number of Tanks	26
Number of Pipes	686
Number of Pumps	32
Number of Valves	39

Although not all of these elements will be subject to optimisation, the complexity involved with this network is clearly much greater than the simple network given above.

In this study, two different problems setups are considered, ranging from highly complex involving all elements of the network to smaller problems which only consider the pipe sizes required in the optimisation of the WDN.

The largest of the problems consists of some 161 decision variables which includes pipe sizes, pump and valve settings and a variety of tank expansion sizes. This variability in the number of decision variables makes the complexity a little more difficult to determine, but the number of options for the algorithms to consider is in the region of $6 *10^{82}$ for this largest problem.

Ant Colony Optimisation Approaches to WDN Optimisation

Ant colony optimisation (ACO) has been successfully applied to a wide range of optimisation problems (Dorigo and Di Caro) and has also been shown to perform very competitively for the optimisation of a WDN (Zecchin et al. 2003). Therefore, existing research shows that ACO and its variants could prove to be a suitable long term alternative to a GA.

According to Simpson et al (2003), ACO can be an appealing alternative to GA for the design of optimal WDNs. In this study ACO was applied to two benchmark WDN optimisation problems and in both scenarios ACO outperformed a GA in terms of computational efficiency and ability to find near global optimal solutions. Additionally Zecchin at al (2007) compared five variations of ACO for the purposes of water distribution network optimisation. The implementations in this study included one basic/standard implementation, an elitist ant system, an elitist-rank ant system (ERAS) and a max-min ant system (MMAS). The results showed that MMAS and ERAS outperformed all other algorithms that have been applied to the same four case studies. The four non standard implementations of ACO are 'current state-of-the-art ACO algorithms that have been applied successfully to variety of combinatorial optimisation problems' (Zecchin et al. 2007). They indicate that the consistently good performance of both ERAS and MMAS makes them stand out from the all other ACO algorithms. It is shown that ERAS is more efficient than MMAS for smaller case studies where as MMAS outperformed ERAS for larger case studies.

Therefore this paper investigates the application of ant colony optimisation to a real-world network taken from industry. The network is far larger than the New York Tunnels example and had previously been optimised using a GA which took several days to complete. The investigation will enable us to determine whether ant colony optimisation can be used to optimise the vastly increased search spaces associated with industrial water distribution networks.

2 METHOD

Infrastructure

The standard method for utilising GAs in WDN optimisation is to couple the GA with a hydraulic simulator such as the one used in this study, Epanet (Rossman, 1999). The GA generates a chromosome representing a candidate solution which is then evaluated in the normal way using the objective function. The objective function calls the DLL and simulates the solution in the Epanet and returns a number of computed results. The function then converts these results into fitness and penalty values and returns an overall fitness to the GA which can then proceed with algorithm.

In this study, we adopted a similar strategy whereby all of the calculations necessary for determining the objective function values were completed within a separate DLL, incorporating the hydraulic solver. The ant colony or any other optimisation algorithm simply passes the DLL the candidate solution and the DLL returns a single value indicating the fitness of that solution. By utilising this information-hiding method, the algorithm can be substituted quite easily with minimal recoding required.

Problem Setup

As described in the introduction, a number of different problems were considered in the study to determine the effect of changes in problem complexity on the algorithms involved. However, this only effects the decisions made by the algorithm. The objective function remained the same for all runs and was as follows:

Cost+G(PenaltyCost)

The G term is to balance the difference in magnitude between the two costs. The cost calculations are quite complex, but effectively each element that the algorithm selects to enhance the network has a known attached cost and these are simply summed together to give the total cost.

The penalty cost relates to the constraints placed on the network and therefore takes into account the following constraints:

- **Required Head Constraints:** All nodes in the network require a certain pressure 'head' to maintain service to customers. This constraint computes the difference between the actual pressure in the node and the required. If the pressure is too low, then the solution is penalised proportionately
- **Tank Level Constraints:** The solution is penalised if a tank drops below a pre-determined threshold or 'overspills'. Additionally, the tank must return to a certain percentage of its original level over 24 hours.
- **Velocity Constraints:** The solution is penalised if the velocity of the water passing through the pipes is too high (leading to an increased likelihood of leakage) or too low (leading to poor water quality).

These constraints are multiplied by constants to increase or decrease their importance to the algorithm and summed together to give the total penalty cost.

The setting of the constants for the objective functions is a non-trivial problem in itself, and the settings used in this study were the result of extensive experimentation with the GA. They were kept constant for both algorithms throughout this study.

Problem Setups

This study includes four problem setups they are as follows:

Setup 1 – considers 161 variables including pipes, pumps, tanks and valves. This is the largest problem and has around 10^{82} possible combinations

Setup 2 – similar to setup 1, but considers only pipe sizing and therefore 121 variables and a complexity in the region of 10^{71}.

ACO Methods

Basic Ant System

BAS calculates the probability of selecting a particular path at any given decision point according to the level of pheromone on that path and (optionally) some local heuristic values.

The calculated probability is then used in a probability proportionate roulette wheel which selects a path. The roulette wheel reinforces good solutions as the option with the highest probability of being selected has proportionately more chance of being selected on the wheel. The element of randomness involved in a roulette wheel encourages exploration and can help avoid stagnation. For more detail information on BAS readers are directed towards Dorigo (1996).

Max-Min Ant System

The MMAS was first proposed by Stützle and Hoos as a variant of the basic ant system. It has also been shown to be an attractive alternative to the GA (Bullenheimer et al, 1997). MMAS follows the same procedure for selecting a path as is described for BAS. MMAS differs to the basic ant system in the way in which the pheromone trails are updated. In MMAS pheromone is only added to one solution per iteration, where a combination of updating the iteration best solution (Sib) and the global best solution (Sgb) is utilised. A slight variation of the MMAS proposed in Stützle and Hoos (2000) is implemented. The difference occurs in the combination of updating the iterations *Sib* and *Sgb*. Similar ratios to Stützle and Hoos are used in this implementation but the change from using *Sib* to *Sgb* is marked by a percentage of the number of iterations opposed to actual values. In this implementation for the first 10% of iterations only the *Sib* is updated, from 10% to 30% update *Sgb* once per 6 iterations, from 30% to 70% update Sgb once per 4 iterations, from 70% update *Sgb* once per 3 iterations. Using the percentage of iterations as a ratio marker opposed to actual values allows the number of iterations to vary.

Elitist-Rank Max-min Ant System

ERMMAS is a new technique that incorporates elements from MMAS and elements from the elitist rank ant system (ERAS) (Bullenheimer et al, 1997). ERMMAS uses dynamic pheromone limits as described for MMAS and an elitist rank update scheme as used in ERAS. Once per iteration ERMMAS awards rank proportionate pheromone to the elite ant's solutions. The focus in ERMMAS shifts from iteration best update to global best update in the same way as MMAS. ERMMAS updates e iteration-elite solutions (Sie) and e global-elite solutions (Sge). ERMMAS offers more efficiency in the use of fitness evaluations than MMAS by updating e trails opposed to 1 trail per iteration. Two different solutions can be virtually identical in cost and penalty cost but comprise very different components. In this scenario MMAS will only reward the solution with the lowest cost ignoring the solution that is virtually as fit. The solution that is not rewarded may however lead to better quality solutions than the solution with the higher fitness. ERMMAS would reward both solutions and thus has a higher probability than MMAS of finding a solution with lower cost and penalty cost.

The pheromone addition equation for ERMMAS is:

$$\Delta\tau_{i(j)}^{(t)} = \begin{cases} \left(\dfrac{Q}{f(S^{ie}(t))}\right) / R & \text{if } i_{(j)} \in S^{ie}(t) \\ 0 & \text{Otherwise} \end{cases}$$

Where R = the rank of the solution; dividing the pheromone addition by the rank ensures that only the solution that is ranked as number 1 will receive the full amount of pheromone. The number of elitist ants e is calculated as a 10th of the population size.

3 RESULTS

Determining Parameter Settings

As with many algorithms, ACO relies on a number of parameter settings being correctly set for the algorithm to function well. In this study, the pheromone decay and population size parameters were experimented with to determine reasonable settings for these. However, the hydraulic solver requires some 0.6 seconds to complete a network evaluation, so these experiments were conducted with only 250 iterations of the ACO algorithms.

Pheremone Decay

The optimum value for the pheromone decay parameter ρ for the standard ant system is previously been shown to be 0.8 (Simpson et al. in 2003) over several case studies. Therefore, this is used as a starting point for experiments performed on BAS. Six experiments are performed with the ρ from 0.7 through to 0.95. These experiments showed that for this case study BAS performs best with $\rho = 0.85$.

Stützle and Hoos (200) showed that with MMAS the lower the ρ the quicker convergence occurs but with a higher likelihood of being in a local maximum and found $\rho = 0.98$ to be optimum for the traveling salesman problem. The example WDN problem is significantly larger than the travelling sales man problem used in that study and therefore due to the size of the problems at hand it was decided to set $\rho = 0.99$ to ensure the slowest convergence possible by applying minimum decay at each iteration. Later experiments were conducted to analyse the effect on the quality of solutions provided when varying ρ.

Population Size

A large population size is not very efficient with respect to the number fitness evaluations as a lot of fitness evaluations are conducted before the trails are updated and used. Several experiments were performed with BAS to test the effect of altering the population size. Four experiments were performed with the population sizes 10, 50, 100 and 1000. It was observed from these experiments that over a given number of fitness evaluations the difference caused by using different population sizes narrows to virtually nothing. A population size of 100 provided solutions with slightly lower cost and penalty cost than all other populations sizes tested however the difference is marginal. For all further experiments with BAS a population size of 100 is used.

A population size of 100 is used in experiments with MMAS however given that MMAS only adds pheromone to one solution per iteration this means that 99 fitness evaluations are not utilised. Experiments have been performed to test the efficiency of using a population size of 60 compared to 100. The results showed that on every occasion a population size of 100 found a final solution with lower cost and penalty cost than a population size of 60 found.

Optimisation Results

Setup 1

Figure 3 plots the total cost of the global best solution f(Sgb) for BAS, MMAS and the GA each time a new global best solution Sgb is found. MMAS initially requires more fitness evaluations than BAS to reach a certain level of fitness. The evolution in BAS tails off at around 80 000 fitness evaluations where a new global best solution is not found from this point forward.

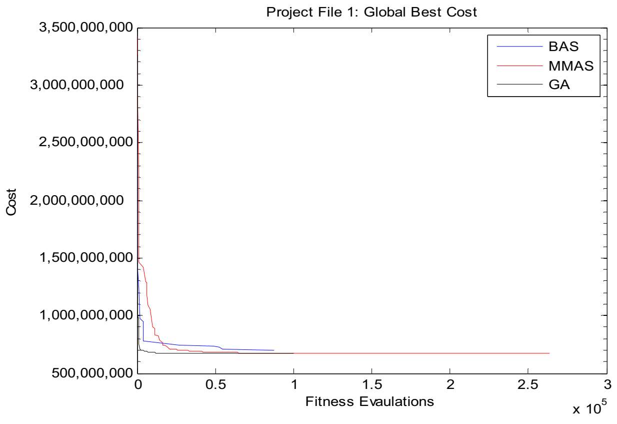


Figure 2 Fitness of the Global Best Solution for BAS, MMAS and the GA on Setup 1

In Figure 2 MMAS and BAS are both run for 300 000 fitness evaluations. MMAS continues to evolve the quality of solutions until over 200 000 fitness evaluations and shows evidence of further improvement. On the final iteration performed by MMAS the difference between the fitness of the iterations best solution f(Sib) and the mean average fitness of solutions f(Φ) was 921,083,790. The difference between the f(Φ) and f(Sib) indicates that further improvement on the quality of solutions could be achieved given more fitness evaluations as close f(Φ) and f(Sib) values indicate that stagnation is occurring. Whereas the greater the difference between f(Φ) and f(Sib) the more exploration is being conducted. MMAS appears to be the algorithm of choice here as it has achieved a much fitter solution than BAS. BAS both improves the quality of solutions slower than the GA and provides a less fit final solution and therefore is poorer in both respects. MMAS has achieved a fitter final solution than the GA but requires more fitness evaluations than the GA.

Up to approximately 50,000 fitness evaluations the GA improves the quality of solutions more quickly than MMAS. From 50,000 fitness evaluations onwards the GA sees little to no improvement while MMAS continues to improve the quality of solutions, eventually providing a fitter solution than the GA.

Table 2 shows the average and best final solution fitnesses and produced by BAS and MMAS for Project File 1. Table 1 shows that MMAS has produced the solution with the lowest f. f for the best solution from MMAS is approximately 1,000,000 less than f for the best solution from the GA and 20,000,000 less than the best solution from BAS. On average MMAS provides a fitter final solution than the GA does for setup 1.
Bas does not perform very well on Project File 1 in terms of ability at finding near optimum solutions. BAS prematurely converges at a local maximum on each of the three runs performed. This is attributed to the non-exclusive approach to pheromone addition causing excessive build of pheromone on particular trails.

Table 2 - Final results for all three algorithms on Setup 1

Method	BAS	MMAS	GA
Average	69617655	673481044	
Best	695982293	672193950	673622800
Fitness Evals	81000	234600	100332

The fitness evaluations row in Table 2 displays the amount of fitness evaluations required by each algorithm to find the best solution.

Setup 2

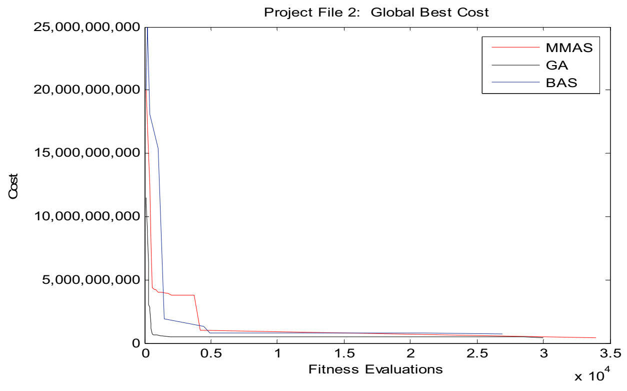


Figure 3 - Fitness of the best solution for BAS, MMAS and the GA using Setup 2

Figure 3 shows a trace of the three algorithms on problem setup 2. As can be seen BAS improves the quality of solutions quicker than MMAS initially but ultimately fails to find a fitter solution than either of the other algroithms. However, BAS sees slower improvement in the fitness of solutions than the GA and results in a less fit final solution. MMAS displays the slowest initial improvement in quality of solutions but proceeds to find the fittest solution overall.

Table 3 shows that MMAS has produced the solution with the lowest fitness but required more fitness evaluations than the GA to achieve this.

Table 3 - Average and best results for all algorithms on Setup 2

	BAS	MMAS	GA
Average	706176556	500905643	
Best	696982293	382092863	448147077
Fitness Evals	25100	34000	30106

Table 3 shows that the solution with the lowest fitness produced by BAS is nearly 300,000,000 higher than the GA. The performance by BAS is significantly worse than the GA and MMAS on these problems and the was made decision not to include BAS in the following experiments due to the consistently poor performance of BAS over Setup 1 and 2.

Accelerating ACO

It has been shown that MMAS is capable of finding solutions with lower fitness than the GA. The focus, therefore, is now on the number of fitness evaluations required. MMAS will be encouraged to converge quicker in the following short experiments by adjusting ρ. Stützle and Hoos (200) showed that a higher ρ value leads to slower convergence and greater chance of finding a near optimum solution therefore, in the following experiments the effect of altering ρ was be examined. It is expected that lowering the value of ρ will initially show quicker improvement in the quality of solutions but will lead to less fit final solutions. These experiments are performed using setup 1 and run for 5000 fitness evaluations.
ERMMAS is included to replace BAS in the following experiments. Figure 4 displays the results of the ρ-value experiments for MMAS and shows that a value of 0.75 for ρ encourages the fastest convergence and a value of 0.99 results in the slowest convergence as expected. These results therefore confirm those found by Stutzle and Hoos (2000).

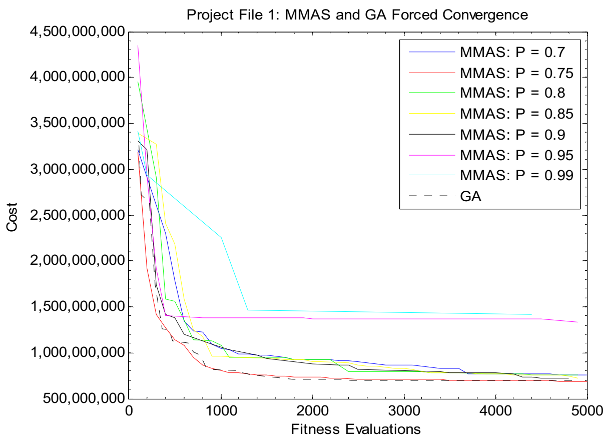


Figure 4 - Comparison of fitness traces for MMAS with varying p-values and the GA for Setup 1.

With ρ = 0.75 MMAS improves the quality of solutions quicker than the GA for the first 400 fitness evaluation and the last 1500 iterations resulting in a fitter solution being found by MMAS. In fact the performance of MMAS with this setting is very close to that of the GA and for many parts of the curve, is better.

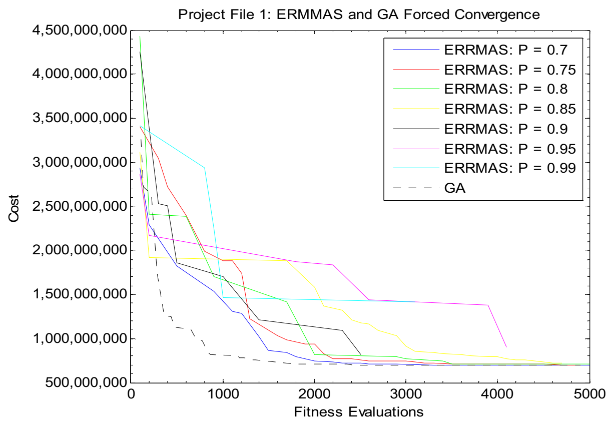


Figure 5 - Comparison of fitness traces for ERMMAS with varying p-values and the GA for Setup 1.

Figure 5 shows that a ρ value of 0.7 has achieved the quickest convergence leading to the fittest solution found by ERMMAS. However, ERMMAS sees significantly slower improvement in the quality of solutions than the GA until around 3500 evaluations. The difference in speed at which the quality of solutions is improved is large when using lower ρ values such as 0.7 or 0.75 compared to ρ = 0.99 for MMAS and ERMMAS.

Table 4 - Comparison of best solution fitnesses all algorithms over 5000 evaluations on Setup 1

P	MMAS	ERMMAS	GA
0.7	712382054	694421933	695426474
0.75	689732784	694555138	
0.8	707236067	703178145	
0.85	704312270	704279302	
0.9	694350970	743975498	
0.95	758040529	906065510	
0.99	1294801273	1058531197	

Table 4 displays Σ and Ω for MMAS and ERMMAS when using problem Setup 1 and a variety a values for the ρ. Table 4 shows that for MMAS a ρ value of 0.9 results in the fittest solution for all the algorithms. Whereas for ERMMAS, a ρ value of 0.7 provided the lowest cost solutions in this shorter timeframe.

4 CONCLUSIONS

In this study we have shown that ACO techniques can be used as effective competitor approaches to genetic algorithms in this important problem domain. The MMAS and ERMMAS ACO algorithms have shown the capability to discover more optimal results that the GA over long-term optimisation runs. This was to a certain extent unexpected as the objective of the study was to show that ACO algorithms could improve on the speed of convergence of the GA, whereas in most cases this wasn't shown. The latter experiments have also shown that the pheromone evaporation rate is crucial to the optimal functioning of the ant system. This is not surprising, but it does highlight the importance of a correctly setup algorithm when dealing with problems of this complexity.

Therefore, while the anticipated computational savings from using ACO did not materialise, some improved results were achieved. There are potentially a variety of reasons why this is the case, including the possibility that the extremely large and rugged search space associated with these problems had a part to play in the slow convergence. It should also be remembered that in the above experiments, the ACO approaches are competing with the best of a number of well-tuned GA runs on the same problem, so the fact that ACO on occasion improves on that result is to its credit. Through this and previous work, it is perhaps becoming apparent that an incremental improvement in WDN optimisation can be achieved through the use of swarm-intelligence based techniques.

An additional advantage to the ACO approach is that it can be more transparent in its decision making that the genetic algorithm. By visualising the pheromone table on the map of possible options, a variety of statistics about the most popular choices for certain components could be generated. Although this has not been investigated in this study, our industrial partners have indicated that this could be a valuable tool in elucidating the decisions being made by the algorithm and therefore increasing confidence in the results.

REFERENCES

[1] Bullnheimer. B., Hartl. R.F., Strauss. C. (1999). A new rank based version of the ant system — a computational study, Central European J. Oper. Res. Econom. 7. pp. 25–38

[2] Beasley. J. E., Chu't'. E C. (1997). A Genetic Algorithm For The Generalised Assignment Problem. Computers Ops Res. Vol. 24, No. 1, Elsevier Science Ltd, pp. 17-23

[3] Dorigo. M., Maniezzo. V., Colorni. A. (1996). The ant system: optimization by a colony of cooperating ants. IEEE Trans. Syst. Man Cybern., 26, pp. 29–42

[4] Dorigo. M., Gambardella. L. (1997). Ant colonies for the traveling salesman problem. BioSystems, 43, pp. 73–81

[5] Dorigo. M., Di Caro. G. (1999). The ant colony optimization metaheuristic. New ideas in optimization. Glover, eds., McGraw-Hill, London, pp. 11–32

[6] Simpson. A. R., Maier. H. R., Foong. W. K., Phang. K. Y. Seah. H. Y., Tan, C. L. (2001). Selection of parameters for ant colony optimization applied to the optimal design of water distribution systems. Proc., Int. Congress on Modelling and Simulation, Canberra, Australia, pp. 1931–1936

[7] Simpson. A. R.., Maier. H. R., Foong. W. K., Phang. K. Y., Seah. H. Y., Tan. C. L. Zecchin. A.C. (2003). Ant Colony Optimization for Design of Water Distribution Systems. Journal of Water Resources Planning and Management, Vol. 129, No. 3, May 1

[8] Stützle. T., Hoos. H.H. (2000). MAX–MIN Ant System. Future Generation Computer Systems 16, Elsevier , pp. 889–914 **able 3:** Forced convergence - Project File 1 varying ρ

[9] Zecchin. A.C., Maier. H.R.., Simpson. A.R., Roberts. A.J. Berrisford. M.J., Leonard. M. (2003). Max-Min Ant System Applied to Water Distribution System Optimisation. In D.A. Post, editor, Proceedings MODSIM 2003: International Congress on Modelling and Simulation, Townsville, Queensland, Australia, July 14–17. pp 795–800

[10] Zecchin. A.C., Maier. H.R.., Simpson. A.R., Leonard. M., Nixon. J.B. (2007). Ant Colony Optimization Applied to Water Distribution System Design: Comparative Study of Five Algorithms. Journal of

Water Resources Planning and Management, Vol. 133, No. 1, January 1

[11] Dandy, G. C., Simpson, A. R., and Murphy, L. J. (1996). An improved genetic algorithm for pipe network optimisation. Water Resour. Res., 32(2), 449–458.

[12] Savic, D. A., and Walters, G. A. (1997). ‘Genetic algorithms for the least-cost design of water distribution networks. J. Water Resour. Plng. and Mgmt., ASCE, 123(2), 67–77.

[13] Rossman, L. A. (1999). Computer models/ EPANET. Water Distribution Systems Handbook. Mays, L. W. ed., McGraw-Hill, New York.

Estimation of Hidden Markov Models Parameters using Differential Evolution

Ângela A. R. Sá[1], **Adriano O. Andrade** [1], **Alcimar B. Soares** [1]

and **Slawomir J. Nasuto** [2]

Abstract. Hidden Markov Models (HMMs) have been successfully applied to different modelling and classification problems from different areas over the recent years. Ann important step in using HMMs is the initialisation of the parameters of the model as the subsequent learning of HMM's parameters will be dependent on these values. This initialisation should take into account the knowledge about the addressed problem and also optimisation techniques to estimate the best initial parameters given a cost function, and consequently, to estimate the best log-likelihood. This paper proposes the initialisation of Hidden Markov Models parameters using the optimisation algorithm Differential Evolution with the aim to obtain the best log-likelihood.

1 INTRODUCTION

Mathematical models are very useful tools for processing and understanding random signals [1, 2]. Hidden Markov Models (HMMs) are statistical models, which have been successfully applied to areas such as speech analysis and biomedical signal processing [1]. An HMM is considered to be an extension of Markov models which models the dynamics of hidden states each of which produces a value of an observable [3].

Although HMMs have been applied in practice, a frequent problem that researchers have to face is that of selecting or estimating initial values for the parameters of the model [4]. It is well known that improperly initialized models lead to inaccurate estimates of the HMM parameters which may be detrimental in subsequent application of the HMM [5].

A number of strategies have been developed to initialize the main parameters of HMMs, such as the application of clustering algorithms (e.g., k-means clustering)[6, 7], Gaussian Mixture Models [6-10], and the use of random values [4, 5, 11-13]. Within the maximum likelihood framework the best initialization leads to the highest likelihood of the data [14].

However, there is not a common consensus concerning the use of any criterion to select the technique to be used. There is no any guarantee that the main parameters initialized to theses strategies will be best one, i.e., that the parameters will be optimized and, consequently, the model will have the highest likelihood.

Therefore, in this work we propose an alternative solution for initialization of the parameters of HMMs. This solution employs Differential Evolution (DE), which is a global optimization algorithm. The standard Differential Evolution (DE) algorithm, belonging to the family of Evolutionary Algorithms, was described by Storn and Price [15, 16]. It is based on evolution of a population of individuals, which encode potential solutions to the problem and traverse the fitness landscape by means of genetic operators that are supposed to bias their evolution towards better solutions. DE is a relatively new optimisation technique compared with other more established Evolutionary Algorithms, such as Genetic Algorithms, Evolutionary Strategy, and Genetic Programming.

The use of DE in the initialisation of HMM parameters alleviates the problems faced by the local techniques, that are usually used for initialisation, and leads to an improved estimation of the maximum log-likelihood parameters.

2 DEFINITION OF THE SET OF PARAMETERS OF HMMs

An HMM is a statistical data analysis method employing probability measures to model sequential data represented by a sequence of observations [17]. Rabiner [11] also defines an HMM as "a doubly embedded stochastic process with an underlying process that is not observable (i.e., it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observations".

A standard HMM has the following parameters [4]:

N = number of states;
$\mathbf{s} = \{S_1, S_2, ..., S_N\}$ – set of (hidden) states;
M = number of symbols;
$\mathbf{v} = \{v_1, v_2, ..., v_M\}$ - set of observations, which may be scalars or vectors;
$\boldsymbol{\pi} = [\pi_1, \pi_2, \pi_3, \ldots, \pi_N]$ – set of initial probability distribution of starting in state S_i at $t = 1$;
$\mathbf{o} = [O_1, O_2, ...O_T]$; $O_i \in V$; $\forall_i$; the observed sequence

[1] Biomedical Engineering Laboratory, Faculty of Electrical Engineering, Federal University of Uberlandia, Brazil

[2] Cybernetics, School of Systems Engineering, University of Reading, Reading, United Kingdom

T = length of the observation sequence (total number of clock times);

t = clock time; t $\in$ (1,2,3,…,T);

A = matrix of transition probabilities a_{ij} for moving from state S_i to state S_j in one time step;

B = {$b_j(k)$} observation probabilities ($b_j(k)$) of observing symbol V_k while being in state S_j.

HMM parameters are often collectively referred to as $\lambda = (\pi, A, B)$. With this set of parameters, it is possible to generate a series of observations, and given the above parameters one can compute the likelihood, $L(o|\lambda)$, of such series by taking the product of the individual probabilities of moving from one state to the next and producing the observations $\mathbf{o}_t$, t = 1,…, T ($\mathbf{o}_t \in$ V) in those states [3]. However, it is a challenging task to initialize HMM parameters, and a proper initialization is essential [4].

3 OPTIMIZATION OF HMM PARAMETERS WITH DIFFERENTIAL EVOLUTION

DE is an optimisation algorithm that creates new candidate solutions by combining the parent individual and several other individuals of the same population. DE is characterized by steps of Genetic Algorithms, referred to as *mutation*, *crossover* and *selection*.

During the initialisation of the algorithm, a population of *NP* vectors, where *NP* is the number of vectors, each of dimension *D* (which is the number of decision variables in the optimisation problem), is randomly generated over the feasible search space. Therefore, the population of DE consists of *NP D*-dimensional parameter vectors $\mathbf{x}_{i,G}$, where $i = 1,2,..., NP$, for each generation *G*.

In the *mutation* step, a difference between two randomly selected vectors from the population is calculated. This difference is multiplied by a fixed weighting factor, *F*, and it is added to a third randomly selected vector from the population, generating the *mutant vector*, **v** [15, 16, 18].

After mutation, the *crossover* is performed between the vector (**x**) and the mutant vector (**v**) (Figure 1), using the scheme in (1) to yield the trial vector **u**. The crossover probability is determined by the crossover constant (*CR*), and its purpose is to bring in diversity into the original population [19].

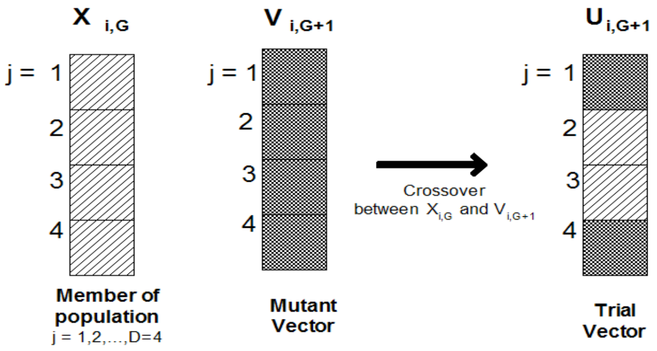


Figure 1. Illustration of the crossover process for D=4.

$$U_{ji,G+1} = \begin{cases} V_{ji,G+1} & \text{if (randb(j)} \le CR \text{ or j = rnbr(i)} \\ X_{ji,G} & \text{if (randb(j)} > CR\text{) or j} \ne \text{rnbr(i)} \end{cases} \quad (1)$$

In (1), *randb(j)* is the jth evaluation of a uniform random number generator with outcome $\in$ [0,1]. *Rnbr(i)* is a randomly chosen index $\in$ 1,2,…,*D*, which ensures that $\mathbf{u}_{i,G+1}$ gets at least one parameter from $\mathbf{v}_{i,G+1}$ [15].

In the last step, called *selection*, the new vectors (**u**) replace their predecessors if they are closer to the global optimum.

The result of DE algorithm will be the *best member* of the population, i.e., the member of the population with the most optimal value of the cost function.

There are two main parameters of HMM that can be optimised using the DE algorithm: the matrices *A* and *B*. These matrices will be optimised until they assume the greatest value, i.e., until that the matrices *A* and *B* generate the best log-likelihood.

In order to optimise the HMM, the first step is to generate a sequence of observations (***o***) from HMM parameters, so that we can estimate their expected log-likelihood. The second step is to apply the DE algorithm to optimise the matrices *A* and *B*, and consequently, to search for the best model that generates the sequence (***o***). Note that these parameters (i.e., A and B matrices) could have a physical meaning, for instance, they could represent an underlying process responsible for the generation of biological signals.

As result of the application of the DE algorithm the matrices *A* and *B* are globally optimised.

4 EXPERIMENTAL PROTOCOL

In order to assess the optimization algorithms, a Hidden Markov process was generated through the HMM Matlab toolbox as illustrated by the following command line:

$$[\texttt{O,states}]=\texttt{hmmgenerate}(\texttt{T},\mathbf{A}_{\texttt{original}},\mathbf{B}_{\texttt{original}})$$

where **o** is a vector of observations, *states* is a vector with the state of each observation, *T* is the total number of observations (O_1, O_2, …, O_T), $\mathbf{A}_{\text{original}}$ is the transition probability matrix and $\mathbf{B}_{\text{original}}$ is the observation probability matrix. Note that when applying HMMs one is interested in obtaining the actual parameters ($\mathbf{A}_{\text{original}}$, $\mathbf{B}_{\textbf{original}}$) that define the rules for data generation.

The genetic operations of *mutation* and *crossover* in DE require the transformations of matrices $\mathbf{A}_{\text{orignial}}$ and $\mathbf{B}_{\text{original}}$ into a one-dimensional vector **ab**, according to (2), (3) and (4).

$$A_{original} = \begin{bmatrix} a_{11} & a_{12} & a_{1N} \\ ... & ... & ... \\ a_{N1} & a_{N2} & a_{NN} \end{bmatrix} = a = [a_{11}, a_{12}, ..., a_{NN}] \quad (2)$$

$$B_{original} = \begin{bmatrix} b_{11} & b_{12} & b_{1M} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ b_{N1} & b_{N2} & b_{NM} \end{bmatrix} = b = [b_{11}, b_{12}, ..., b_{NM}] \quad (3)$$

$$ab = [a_{11}, a_{12}, ..., a_{NN}, b_{11}, b_{12}, ..., b_{NM}] \quad (4)$$

The parameters of the DE algorithm were defined as follows: *D* = length of the vector **ab**, *CR* = 0.8 (*crossover* constant), *F* = 0.8 and *NP* = 100 (number of members of the population), with a fixed number of 1000 generations (iterations) and 1000 simulations.

Two types of HMM models were considered in the analysis:

- An HMM model with two states (N=2) and two symbols (M=2);
- An HMM model with three states (N=3) and four symbols (M= 4).

To evaluate the results, for each type of model, we calculated the data log-likelihood using the parameters estimated by the *best member* of the population and also considering the average over all members of the population.

In order to compare the results obtained with different strategies for parameter estimation, it was generated a graphic that presents the mean of the log-likelihood of the model given the parameters **A** and **B**. Six distinct strategies were considered for the estimation of the parameters **A** and **B**. They are described as follows:

1) *Original matrix*
   The matrices **A** and **B** were set to $\mathbf{A}_{\text{original}}$ and $\mathbf{B}_{\text{original}}$ respectively. Note that these are the actual parameters employed for generation of the sequence of observations;
2) *Original matrix + BW*
   The model was initialised with the matrices **A** and **B** set to $\mathbf{A}_{\text{original}}$ and $\mathbf{B}_{\text{original}}$ respectively, and then optimised by using the Baum-Welch algorithm (BW);
3) *Random*
   The matrices **A** and **B** were initialised at random.
4) *Random + BW*
   The matrices **A** and **B** were initialised at random and then optimised by means of the BW algorithm;
5) *DE*
   The matrices **A** and **B** were optimised by the DE algorithm;
6) *DE + BW.*
   The matrices **A** and **B** were optimised by DE and then optimised by BW;

The log-likelihood of the model given the input sequence **o** was estimated through the Matlab toolbox as depicted below.

```
[log_likelihood]=hmmdecode(A,B,o)
```

The confidence interval for the mean of the likelihood was calculated by using the technique Bootstrap.

5 RESULTS

5.1. HMM model with two states (N=2) and two symbols (M=2)

In this case, it was generated a Hidden Markov process model with two states (N=2) and two symbols (M=2); generated through the HMM Matlab toolbox, as described in the previous section. The vector of observations **o** was generated with *T* equals to 100. The same observations data **o** was used for every 6 distinct strategies described in experimental protocol.

Figure 2 presents the log-likelihood of the data for the 6 distinct strategies of model estimation described in section 4. In the case of *DE* and *DE+BW*, the mean, over 1000 simulations, of the log-likelihood of the *best member* of the population is presented,. The vertical bars represent the confidence interval for the mean.

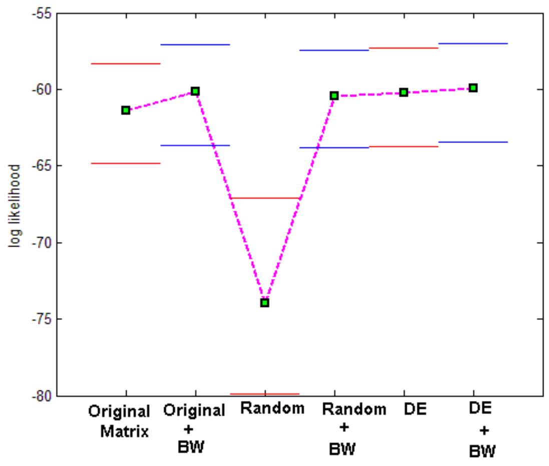


Figure 2. Log-likelihood of the data given the estimated model for 6 strategies of estimating the parameters **A** and **B**. In the case of the DE algorithm **A** and **B** were selected as the best member of the population. The horizontal bars are the upper and lower confidence interval for the mean.

Figure 3 presents for comparison the results obtained for *DE* and *DE+BW*, averaged over all members of the final population (all methods averaged over 1000 simulations). The vertical bars represent the confidence interval for the mean.

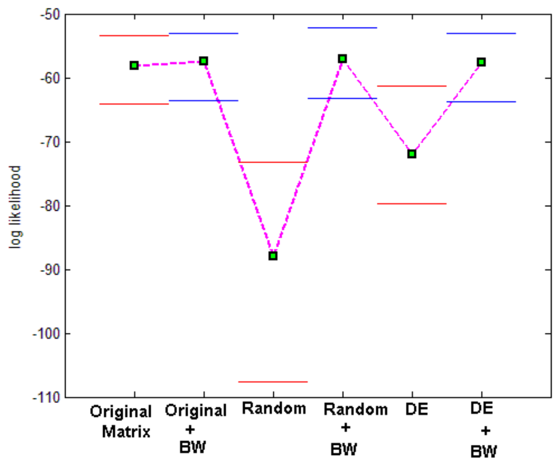


Figure 3 Log-likelihood of the data given the estimated model for 6 strategies of estimating the parameters **A** and **B**. In the case of the DE algorithm, the matrices **A** and **B** were considered as the average of all members of the population. The horizontal bars are the upper and lower confidence interval for the mean.

5.2. HMM model with three states (N=3) and four symbols (M=4)

In this case, it was generated a Hidden Markov process model with three states (N=3) and four symbols (M=4); generated through the HMM Matlab toolbox, as described in the previous section. The vector of observations **o** was generated with *T* equals to 100. The same observations data **o** was used for every 6 distinct strategies described in experimental protocol.

Figure 4 presents the log-likelihood of the data for the 6 estimiation strategies. In the case of DE and DE+BW, the mean, over 1000 simulations, of the log-likelihood of the *best member* of the population is presented.. The vertical bars represent the confidence interval for the mean.

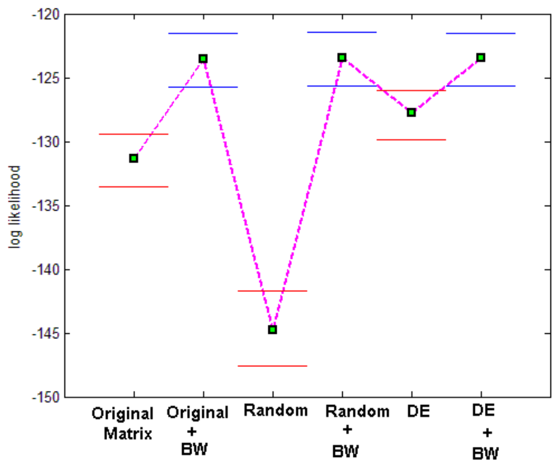


Figure 4. Log-likelihood of the data given the estimated model for 6 strategies of estimating the parameters **A** and **B**.. In the case of the DE **A** and **B** were selected as the best member of the population. The horizontal bars are the upper and lower confidence interval for the mean.

In Figure 5, the results for DE and DE+BW, represent the mean (over 1000 simulations) of the log-likelihood of the average over all members of the population. The vertical bars represent the confidence interval for the mean.

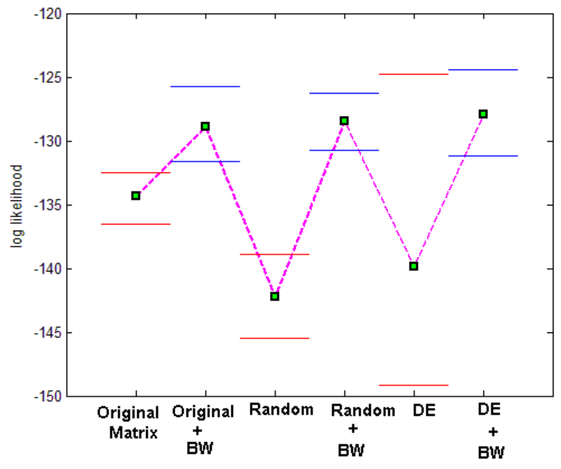


Figure 5. Log-likelihood of the data given the estimated model for 6 strategies of estimating the parameters **A** and **B**.. In the case of the DE, the matrices **A** and **B** were considered as the average of all members of the population. The horizontal bars are the upper and lower confidence interval for the mean.

6 DISCUSSION

From the results presented in section 5, it is possible to compare the behaviour of the 6 distinct strategies for initialisation of the HMM parameters, as described in section 4.

In all cases analysed, the log-likelihood of the *Original Matrix* was lower than all the other optimisation strategies (*Original Matrix + BW*, *DE* and *DE+BW*). This occurs because the matrices $\mathbf{A}_{\text{original}}$ and $\mathbf{B}_{\text{original}}$ generated the sequence **o**, however it does not mean that these matrices are the best ones to generate this sequence, i.e., they are only close to the optimal one. As small is the Euclidian Distance between the estimated matrices and the "best" matrices, given a sequence of observables (**o**), larger will be the log-likelihood of the estimated matrices to the sequence **o**. So, if an optimisation algorithm is applied to these matrices, it is possible to find the best solution suitable and, consequently, they will have a larger log-likelihood. Note that when there is a smaller number of states and symbols, the optimisation *Original+BW* obtains a larger log-likelihood.

In the case of the strategy *Random*, in all simulations it had the lowest log-likelihood. This is explained by the fact that the matrices generated in this strategy are random, and its probability of generating the sequence **o** is small and, consequently, its log-likelihood is always the lowest. But if the BW algorithm is applied to this strategy (i.e., *Random+BW*), it can obtain a log-likelihood larger than the *Original Matrix* and as good as the other optimisation strategies.

It is possible to verify that the initialisation of HMM parameters with DE works as good as the other optimisation strategies when we have a lower number of states (*N*) and symbols (*M*). However, when we have a larger number of states and symbols, the behaviour of DE algorithm can be improved if it is optimised by BW algorithm (*DE+BW*).

In *DE* algorithm, the log-likelihood of the *best member* of the population showed to be always larger than the log-likelihood estimated of the average of all population. However, when the DE is optimised by BW (*DE+BW*) the result of the log-likelihood is as good as the other optimisation strategies (*Original + BW* and *Random + BW*), even considering the best member or the average of the population.

It is already known that the optimisation of the parameters of HMM is a multimodal problem, i.e., it has many local maxima and many local minima. If during the optimisation the parameters get stuck in a local optimum this implies that the model will not have the highest log-likelihood that it could have. When we use a standard approach of optimisation, such as Baum Welch that is a local technique, we will always have the risk of the HMM parameters get stuck in a local optimum. Therefore, according the results depicted above, an interesting alternative to avoid this possible undesirable behaviour of a local optmiser is the use of the global optimiser Differential Evolution.

7 CONCLUSIONS

This paper proposes the use of DE algorithm in the initialisation of HMM parameters. The improperly initialized models lead to inaccurate estimates of the HMM parameters [5].

According to results discussed in section 6, it is possible to conclude that DE algorithm improves the initialisation of the model. When the DE algorithm is compared to the random initialisation optimised, in general, they present a similarity in their results.

It is import to remark that the optimisation of the parameters of HMM is a multimodal problem. Therefore, we have to be aware to avoid that the parameters get stuck in a local optimum and, consequently, to get the highest log-likelihood from its parameters. According to results presented in section 5, the use of the global optmiser DE algorithm, can be important to help the optimisation of HMM parameters do not get stuck in a local optimum and to obtain its best values.

Therefore, the optimisation method proposed in this paper can be aggregated to the different types of strategies that can be successfully utilized to initialize HMM models.

REFERENCES

1. Cohen, A., *Hidden Markov models in biomedical signal processing.* 20th Annual International Conference of the IEEE engineering in medicine and biology society, 1998. **3**.
2. Laverty, W.H., M.J. Miket, and I.W. Kelly, *Simulation of hidden Markov Models with Excel.* The Statistician, 2002. **51**: p. 31-40.
3. Visser, I., M.E.J. Rijamakers, and P.C.M. Molenaar, *Confidence intervals for hidden markov model parameters.* British Journal of Mathematical and Statistical Psychology, 2000. **53**: p. 317-327.
4. Rabiner, L.R., *A tutorial on hidden markov models and selected applications in speech recognition.* IEEE, 1989. **77**: p. 257-286.
5. Nathan, K., A. Senior, and J. Subrahmonia. *Initialization of hidden markov models for unconstrained on line handwriting recognition.* in *Acoustics, Speech, and Signal Processing*. 1996. Atlanta.
6. Panuccio, A., M. Bicego, and V. Murino, *A hidden markov model based approach to sequential data clustering*, in *Structural, Syntactic and statistical pattern recognition*, T. Caelli and A. Amin, Editors. 2002, Springer. p. 734-742.
7. Jung, K.C., S.M. Yoon, and H.J. Kim, *Continuous HMM applied to quantization of on line Korean character spaces.* Pattern Recognition Letters, 2000. **21**: p. 303-310.
8. Nefian, A.V. and M.H. Hayes. *Face detection and recognition using hidden markov models.* in *International Conference on Image Processing*. 1998.
9. Reyes-Gomez, M.J. and D.P.W. Elli. *Selection, parameter estimation and discriminative training of hidden markov models for generic acoustic modeling.* in *International conference on multimedia and expo*. 2003.
10. Kim, D.K. and N.S. Kim, *Maximum a posteriori adaptation of HMM parameters based on speaker space projection.* Speech Communication, 2004. **42**: p. 59-73.
11. Rabiner, L.R. and B.H. Juang, *An introduction to hidden markov models.* IEEE ASSP Magazine, 1986: p. 4-16.
12. Smyth, P., *Clustering Sequences with Hidden Markov Models*, in *Advances in Neural Information Processing Systems*, M.C. Mozer, Editor. 1997, MIT Press.
13. Davis, R.I.A. and B.C. Lovell, *Comparing and evaluating HMM ensemble training algorithms using train and test condition number criteria.* Pattern Anal Applic, 2003. **6**: p. 327-336.
14. Seidemann, E., et al., *Simultaneously recorded single units in the frontal cortex go through sequences of discrete and states in monkeys performing a delayed localization task.* The Journal of Neuroscience, 1996. **2**(16): p. 752-768.
15. Price, K.V., R.M. Storn, and J.A. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*. 2005: Springer. 538.
16. Storn, R. and K. Price, *Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces.* Journal of Global Optimization, 1997. **11**: p. 341-359.
17. Mohamed, M.A. and P. Gader, *Generalized Hidden Markov Models - Part I: Theoretical Frameworks.*

IEEE Transactions on fuzzy systems, 2000. **8**: p. 67-81.

18. Li, X. *Efficient Differential Evolution using Speciation for Multimodal Function Optmization*. in Conference on Genetic and Evolutionary Computation. 2005. Washington DC.

19. Roger, L.S., M.S. Tan, and G.P. Rangaiah, Global Optimization of Benchmark and Phase Equilibrium Problems Using Differential Evolution. 2006, National University of Singapore: Singapore.

Exploration vs. Exploitation in Differential Evolution

Ângela A. R. Sá[1], Adriano O. Andrade[1], Alcimar B. Soares[1]

and Slawomir J. Nasuto[2]

Abstract. Differential Evolution (DE) is a tool for efficient optimisation, and it belongs to the class of evolutionary algorithms, which include Evolution Strategies and Genetic Algorithms. DE algorithms work well when the population covers the entire search space, and they have shown to be effective on a large range of classical optimisation problems. However, an undesirable behaviour was detected when all the members of the population are in a basin of attraction of a local optimum (local minimum or local maximum), because in this situation the population cannot escape from it. This paper proposes a modification of the standard mechanisms in DE algorithm in order to change the exploration vs. exploitation balance to improve its behaviour.

1 INTRODUCTION

The standard Differential Evolution (DE) algorithm, belonging to the family of Evolutionary Algorithms, was described by Storn and Price [1, 2]. It is based on evolution of a population of individuals, which encode potential solutions to the problem and traverse the fitness landscape by means of genetic operators that are supposed to bias their evolution towards better solutions. DE is a relatively new optimisation technique compared with other more established Evolutionary Algorithms, such as Genetic Algorithms, Evolutionary Strategy, and Genetic Programming [3].

DE is an optimisation algorithm that creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness [3, 4]. DE uses genetic operators, referred to as *mutation*, *crossover* and *selection*. The role of the genetic operators is to ensure that there is sufficient pressure to obtain even better solutions from good ones (exploitation) and to cover sufficiently the solution space to maximise the probability of discovering the global optimum (exploration).

During the initialisation of the algorithm, a population of *NP* vectors, where *NP* is the number of vectors, each of dimension *D* (Which is the number of decision variables in the optimisation problem), is randomly generated over the feasible search space.

Therefore, the population of DE consists of *NP* *D*-dimensional parameter vectors $\mathbf{X}_{i,G}$, where $i = 1,2,..., NP$, for each generation *G*.

In the *mutation* step, a difference between two randomly selected vectors from the population is calculated. This difference is multiplied by a fixed weighting factor, *F*, and it is added to a third randomly selected vector from the population, generating the *mutant vector*, **V** [1-3].

After mutation, the *crossover* is performed between the vector (**X**) and the mutant vector (**V**) (Figure 1), using the scheme in (1) to yield the trial vector **U**. The crossover probability is determined by the crossover constant (*CR*), and its purpose is to bring in diversity into the original population [5].

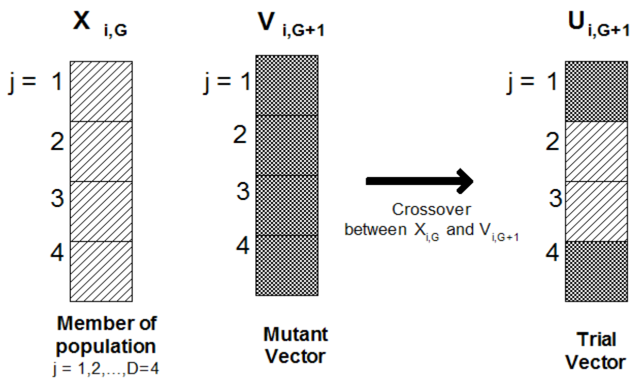


Figure 1. Illustration of the crossover process for D=4.

$$U_{ji,G+1} = \begin{cases} V_{ji,G+1} & \text{if (randb(j)} \le CR \text{ or j = rnbr(i)} \\ X_{ji,G} & \text{if (randb(j)} > CR \text{ or j} \ne \text{rnbr(i)} \end{cases} \quad (1)$$

[1] Biomedical Engineering Laboratory, Faculty of Electrical Engineering, Federal University of Uberlandia, Brazil

[2] Cybernetics, School of Systems Engineering, University of Reading, Reading, United Kingdom

In (1), *randb(j)* is the jth evaluation of a uniform random number generator with outcome $\in [0,1]$. *Rnbr(i)* is a randomly chosen index $\in 1,2,\ldots,D$, which ensures that $\mathbf{U}_{i,G+1}$ gets at least one parameter from $\mathbf{V}_{i,G+1}$ [1].

There are different variants that can be used in *mutation* and *crossover*, and they are referred to as DE/x/y/z, where x specifies the vector to be mutated which currently can be "rand" (a randomly chosen population vector) or "best" (vector of the lowest cost from the current population); y is the number of difference vectors used and z denotes the crossover scheme [2].

In the last step, called *selection*, the new vectors $(\mathbf{U})$ replace their predecessors if they are closer to the target vector.

DE has shown to be effective on a large range of classical optimisation problems, and it showed to be more efficient than techniques such as Simulated Annealing and Genetic Algorithms [4, 5]. However, its capability of finding the global optimum is very sensitive to the choice of the control variable *F* and *CR* [6]. Consistently with related studies [3, 5, 6], the paper highlights an undesirable behaviour of the algorithm, i.e., the DE does not find the global optimum (value to reach - *VTR*) when 100% of the population is trapped in a basin of attraction of a local optimum.

-

2 THE BEHAVIOUR OF THE STANDARD DIFFERENTIAL EVOLUTION

In order to verify the behaviour of standard DE algorithm, the Rastrigin's function defined in (2), representing multimodal optimisation problem, was selected as the cost function. As Figure 2 shows, Rastrigin´s function has many local minima and maxima, making it difficult to find the global optimum. Consequently, it is a very good function to test the behaviour of DE algorithm.

$$f(x,y) = \left(20 + x^2 + y^2\right) + 10 \times \left(\cos(2\pi x) + \cos(2\pi y)\right) \quad (2)$$

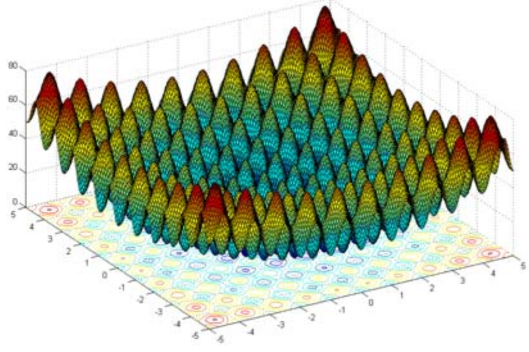


Figure 2. The Rastrigin's Function employed to assess the modification in DE algorithm.

We considered the Rastrigin function on the search domain Dom={(x,y): $-5 \le x \le 5$ and $-5 \le y \le 5$ }. The Rastrigin´s Function has in Dom the global minimum at $(x,y) = (0,0)$, with $VTR = f(x,y) = 0$; and it also has four global maxima at $(x,y) = (4.5,4.55)$, $(x,y) = (-4.5,4.55)$, $(x,y) = (4.5,-4.55)$ and $(x,y) = (-4.5,-4.55)$, with $VTR = f(x,y) = 80.46$. Henceforth, global minimization and maximization of the Rastrigin function respectively represent different levels of difficulty for swarm intelligence algorithms due to non-uniqueness of the global optima in the latter. We will report results of both cases.

The parameters of the DE were defined as follows: D = 2, CR = 0.8 (cross over constant), F = 0.8 and NP = 100 (number of members of population), with a fixed number of 1000 generations (iterations) and 1000 simulations, each one with a different data set.

The strategy used in the *mutation* and *crossover*, i.e., the variants of creation of a candidate, was the DE/rand to best/1, because according to previous simulations, it is this strategy that resulted in a larger number of members that found the global optimum [5, 6].

2.1 Strategy for data analysis

In order to verify the performance of the optimisation of DE algorithm, we executed it for each case, i.e., minimization and maximization.

The results were obtained through three types of analysis that are described as follows:

1) *Estimate of the average number of members of the population, over 1000 simulations, which reached a pre-defined neighbourhood of the VTR* defined in terms of the maximal accepted difference between *VTR* and the population members cost values. The confidence interval for the mean of the frequency by using the technique Bootstrap [7-9] was also estimated.

2) *Contour plot* that shows the initial distribution of the population before and after optimisation.

3) For each *neighbourhood* the average number of occurrence of member of the population, over 1000 simulations, is estimated as a function of the number of iterations.

2.2 Minimization

Minimisation is the simpler case of optimisation due to the uniqueness of the global optimum and is discussed first. Figure 3 shows a typical result of a simulation using the standard DE algorithm. In the top, the initial distribution of the population (in red) is shown; all members of the population are trapped in the local optimum. The *VTR* is indicated in the figure. After optimisation, the members of the population did not escape from the local optimum as depicted in Figure 3 (bottom).

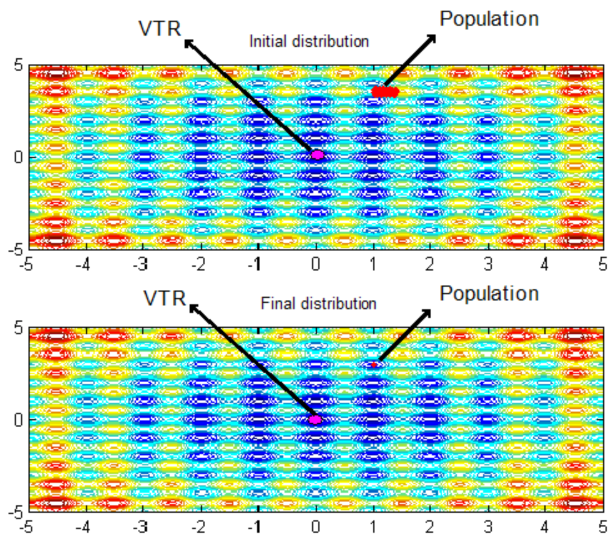


Figure 3. (Top) Initial distribution of the population. **(Bottom)** Final distribution of the population, after running the standard DE algorithm. The arrows in the figure indicate the VTR, the initial and final distribution of the population.

Figure 4 presents the results of applying the standard DE algorithm. The average number of members of the population that reached the *VTR* was zero for all investigated neighbourhoods. This means that the algorithm was not capable of moving the population to the *VTR*, as can be visualized by the typical case shown in Figure 4.

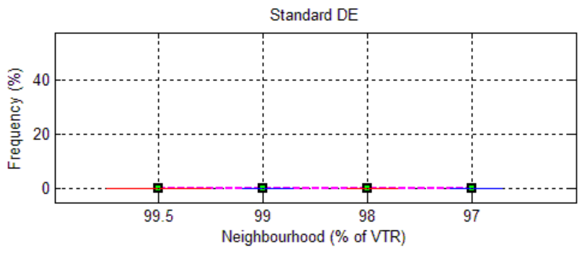


Figure 4. Estimate of the average number of the members of the population, over 1000 simulations, which reached the VTR. These results were obtained from the standard DE algorithm.

These results illustrate the limitation of the standard DE genetic operators which do not create sufficient diversity within the population in order to explore other regions of the search space beyond the basin of attraction of the local optimum.
The more difficult case of global optimisation (due to non-uniqueness of the global maxima of the Rastrigin function) are considered in the next section.

2.3 Maximization

Figure 5 shows a typical result of a simulation using the standard *DE* algorithm. In the top, the initial distribution of the population (in red) is shown; all members of the population are trapped in the local optimum. The *VTR* is indicated in the figure. After optimisation the members of the population did not escape from the local optimum as depicted in Figure 5 (bottom).

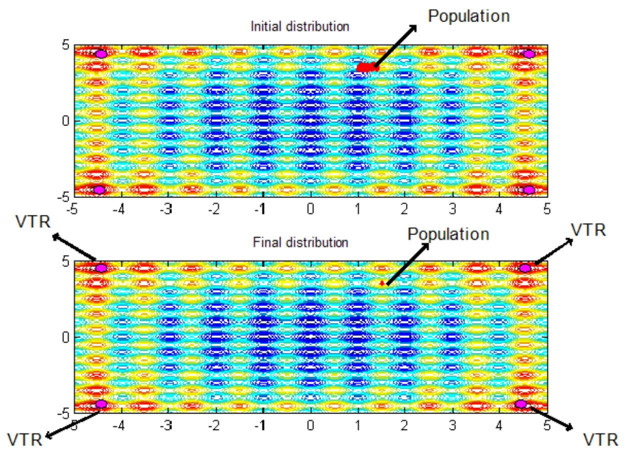


Figure 5. (Top) Initial distribution of the population. **(Bottom)** Final distribution of the population, after running the standard DE algorithm. The arrows in the figure indicate the VTR, the initial and final distribution of the population.

Figure 6 presents the results of applying the standard DE algorithm. The average number of members of the population that reached the *VTR* was zero for all investigated neighbourhoods. This means that the algorithm was not capable of moving the population to the *VTR*, as can be visualized by the typical case shown in Figure 6.

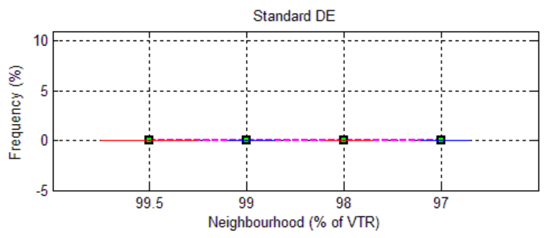


Figure 6. Estimate of the average number of the members of the population, over 1000 simulations, which reached the VTR. These results were obtained from the standard DE algorithm.

As expected, also in this case the standard genetic operators do not result in sufficient exploration of the search space. Bias towards exploitation still results in entrapment in the local optimum.

3 PROPOSED SOLUTION

In order to improve the balance between the exploration and exploitation in DE algorithm we propose a modification of the selection used in DE. The aim of the modification is to restore balance between exploration and exploitation affording increased probability of escaping basin of attraction of local optima.

In the standard DE algorithm, the selection of the new population is made by using a determinist condition, i.e., usually, in the DE algorithm, the *selection* is made by choosing the member (**X** or **V**) that is the nearest to the *VTR* [5]. Because the mutation and crossover are well suited to support efficient exploitation we feel that relaxing this greedy selection scheme will address the identified problem without detrimental effects for the exploitation. We propose to relax the deterministic selection to maintain the population diversity and consequently to allow escape from basin of attraction of local optima. The diversity will reduce the selection pressure which will alleviate the problem caused by greediness of the other operators in cases when most (or all) of the population members are trapped in specific regions of the search space.

A proposed solution is to introduce a probabilistic selection of the new member of the population, in order to induce the population diversity. We define the probability of selection of the population members in terms of their cost function. Hence the probability of selecting **X** is defined in (3) and the probability of **V** in (4).

$$P(X) = \frac{\left(f(X)+K\right)}{\left(f(X)+K\right)+\left(f(V)+K\right)} \quad (3)$$

$$P(V) = 1 - P(X) \quad (4)$$

The probability of selecting X is an increasing function of the cost function evaluated at **X**. The constant, *K*, in (3) is added in order to guarantee that P(**X**) is greater that zero. The value of *K* will depend on the cost function employed in the algorithm.

The probabilistic selection proposed above still maintains some pressure on the population to move towards more optimal solutions but nevertheless allows for greater diversity by admitting sub optimal choices, which are needed to shift the population across the boundaries of the basins of attraction of local optima. The implementation of the probabilistic selection is straightforward by sampling a random number, *c*, from a uniform distribution on a unit interval and comparing with the probability of selecting **X**. The minor modifications needed for the case of minimization and maximization respectively are highlighted below.

- Minimization - We have to choose the member of the population with the lowest value of the cost function. The pseudo-code shown in Figure 7 illustrates this process.

```
If (P(V) < c)
    then U = V;
    else U = X;
```

Figure 7. The probabilistic *selection* stage of the DE for minimization.

- Maximization - We have to choose the member of the population with the highest value of the cost function. The pseudo-code shown in Figure 8 illustrates this process.

```
If (P(V) > c)
      then U = V;
      else U = X;
```

Figure 8. The probabilistic *selection* stage of the DE for maximization.

The original code of DE was obtained from [10], with the original code written by Rainer Storn and Kenneth Price [1, 2], and the modification described above was incorporated on it.

5 RESULTS

In order to test the modifications proposed in the DE algorithm, the same experimental protocol, described in section 2, was used. The results presented below the behaviour of the modified DE on global minimization of the Rastrigin function. The more difficult case (due to non-uniqueness of global optima) of global maximization of this function is discussed subsequently.

5.1 Minimization

Figure 9 shows a typical result of a simulation using the DE algorithm with probabilistic selection. In the top, the initial distribution of the population (in red) is shown; all members of the population are trapped in the local optimum. The VTR is indicated in the figure. After optimisation the members of the population escaped from the local optimum as depicted in Figure 9 (bottom).

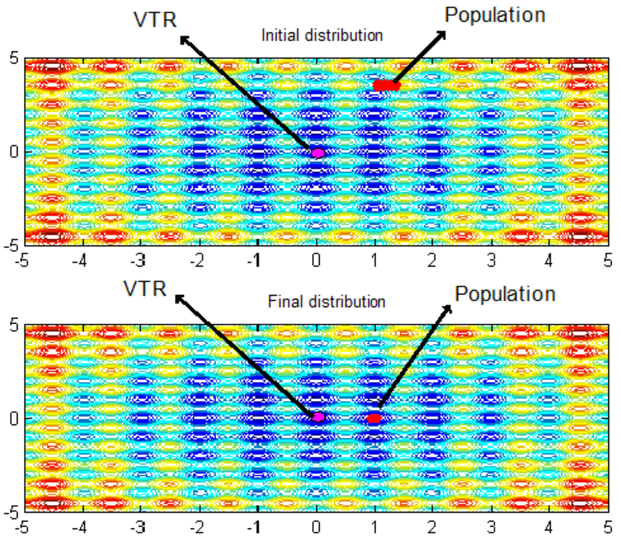


Figure 9. (Top) Initial distribution of the population. **(Bottom)** Final distribution of the population, after running the DE algorithm with probabilistic selection. The arrows in the figure indicate the VTR, the initial and final distribution of the population.

Figure 10 presents the estimate of the concentration of the DE population around the VTR. The vertical bars represent the confidence interval for the mean. The average number of members of the population that reached the VTR was zero for neighbourhoods equals to 99.5% and it was larger than zero for the others. This means that the algorithm is capable of moving the population to a region closer to the VTR.

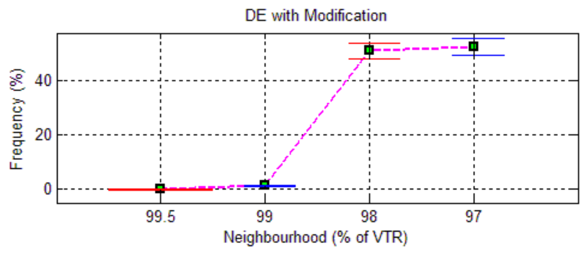


Figure 10. Estimate of the average number of the members of the population, over 1000 simulations, which reached the VTR for the DE with probabilistic selection.

Figure 11 presents the comparison between the average number of members that reached the *neighbourhood* 99% of VTR as a function of time (iterations), for by DE with standard and probabilistic selection. It thus visualises the convergence of the populations towards the VTR during evolution of both algorithms.

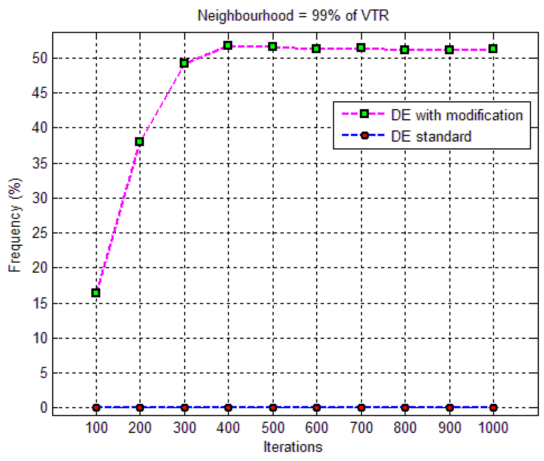


Figure 11. Average size of the population of members reaching the neighbourhood 99% of VTR, as a function of time (iterations).

Figure 12 presents the comparison between the average number of members that reached the *neighbourhood* 98% of VTR as a function of time (iterations), by DE with standard and probabilistic selection.

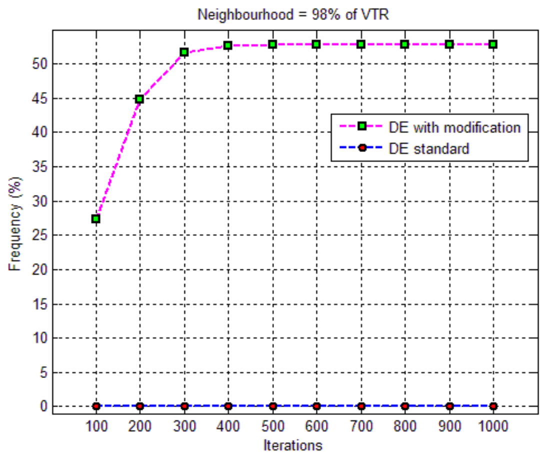


Figure 12. Average size of the population of members reaching the neighbourhood 98% of VTR, as a function of time (iterations).

Figure 13 presents the comparison between the average number of members that reached the *neighbourhood* 97% of VTR as a function of time (iterations), by DE with standard and probabilistic selection.

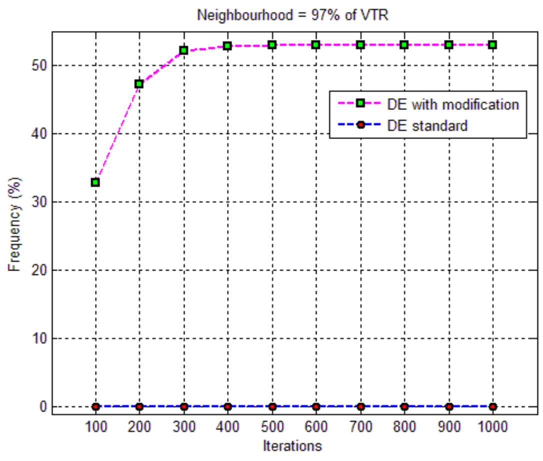


Figure 13. Average size of the population of members reaching the neighbourhood 97% of VTR, as a function of time (iterations).

5.2 Maximization

Figure 14 shows a typical result of a simulation using the DE algorithm with probabilistic selection. In the top, the initial distribution of the population (in red) is shown; all members of the population are trapped in the local optimum. The VTR is indicated in the figure. After optimisation the members of the population escaped from the local optimum as depicted in Figure 14 (bottom).

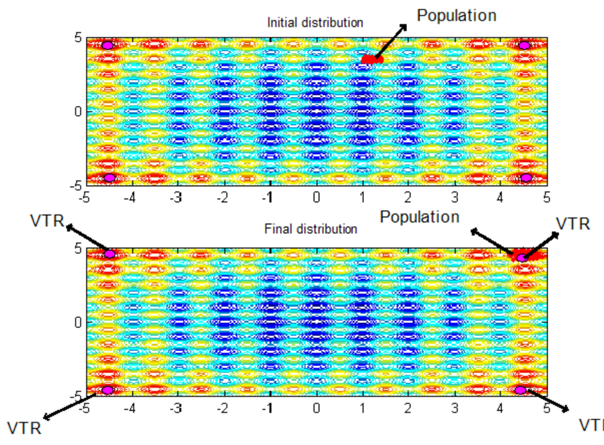


Figure 14. (Top) Initial distribution of the population. **(Bottom)** Final distribution of the population, after running the DE algorithm with probabilistic selection. The arrows in the figure indicate the VTR, the initial and final distribution of the population.

Figure 15 presents the estimate of the concentration of the DE population around the VTR. The vertical bars represent the confidence interval for the mean. The average number of members of the population that reached the VTR was zero for neighbourhoods equals to 99.5% and it was larger than zero for the others. This means that the algorithm is capable of moving the population to a region closer to the VTR.

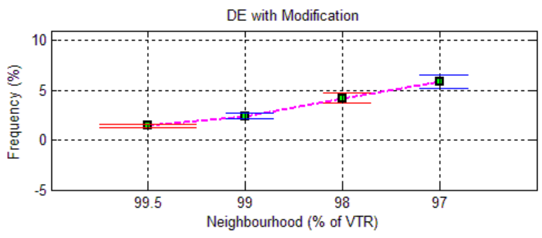


Figure 15. Estimate of the average number of the members of the population, over 1000 simulations, which reached the VTR for the DE with probabilistic selection.

Figure 16 presents the comparison between the average number of members that reached the *neighbourhood* 99% of VTR as a function of time (iterations), for by DE with standard and probabilistic selection. It thus visualises the convergence of the populations towards the VTR during evolution of both algorithms.

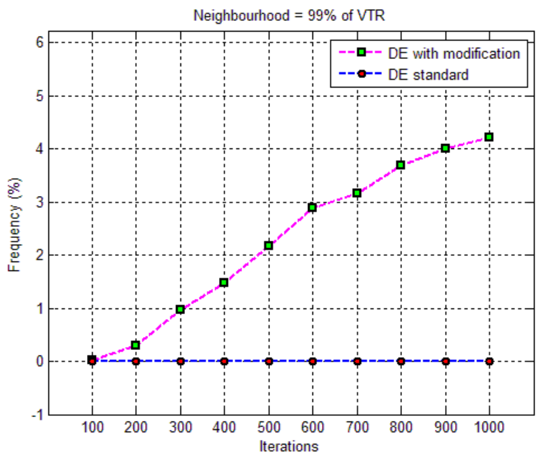


Figure 16. Average size of the population of members reaching the neighbourhood 99% of VTR, as a function of time (iterations).

Figure 17 presents the comparison between the average number of members that reached the *neighbourhood* 98% of VTR as a function of time (iterations), by DE with standard and probabilistic selection.

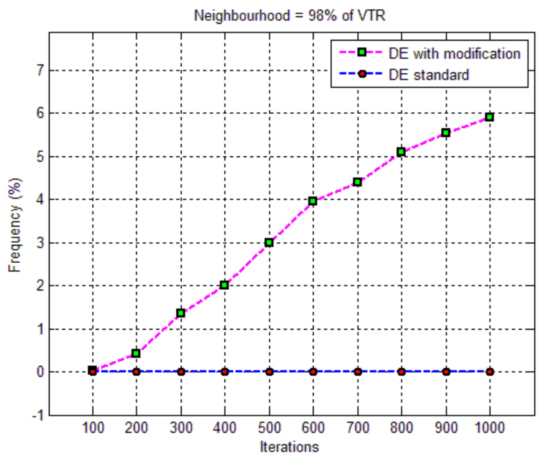


Figure 17. Average size of the population of members reaching the neighbourhood 98% of VTR, as a function of time (iterations).

Figure 18 presents the comparison between the average number of members that reached the *neighbourhood* 97% of VTR as a function of time (iterations), by DE with standard and probabilistic selection.

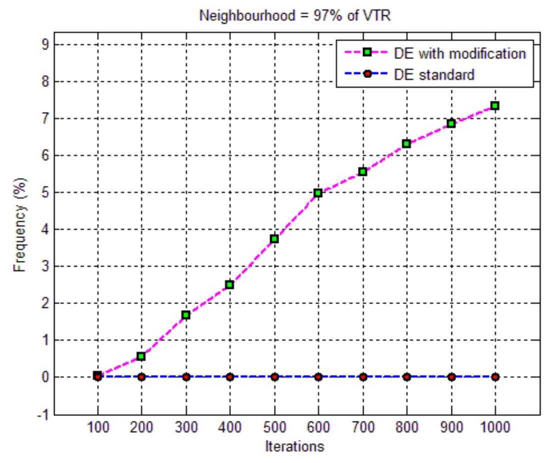


Figure 18. Average size of the population of members reaching the neighbourhood 97% of VTR, as a function of time (iterations).

6 DISCUSSION

The results reported in this paper indicate that alterations of the DE *selection* step improve the optimisation process.

The introduced changes affect the exploration vs. exploitation balance maintained by the DE population. Local exploitation afforded to by standard mutation and crossover combined with the less greedy probabilistic selection allowing for occasional sub optimal choices led to an improved diversity of the population which allowed the modified DE to escape from initial local optima.

In the case of minimization, some members of the modified DE population reached a neighbourhood of the global minimum. It is important to note that all member of the population escaped from the initial local optimum in which they were trapped. The standard DE did not allow the population to escape from the local optimum in the same case.

Similar contrasting behaviour of the two algorithms was observed in the case of maximization. It is important to note that the Rastringin´s Function (Figure 2) has four global maxima and the standard DE algorithm failed to locate them. In contrast the modified DE was able to successfully locate one global optimum.

7 CONCLUSIONS

This paper proposes an extension of the standard DE algorithm with the aim of improving its maintenance of population diversity by restoring more balance between exploration and exploitation of the search space. The modification proposed showed to be successful in avoiding entrapment in local optima.

The modification proposed is important because in multimodal optimization problems even swarm intelligence algorithms have difficulty with escaping the local optima. The probability of success afforded by the modified DE showed to be higher than for the standard algorithm.

According to Rainer and Price [2], little is known about DE behavior in real world applications. This work contributes to the investigation of DE behaviour and the modification proposed in the algorithm can be very useful in many real world optimisation problems.

REFERENCES

1. Price, K.V., R.M. Storn, and J.A. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*. 2005: Springer. 538.
2. Storn, R. and K. Price, *Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces.* Journal of Global Optimization, 1997. **11**: p. 341-359.
3. Li, X. *Efficient Differential Evolution using Speciation for Multimodal Function Optmization*. in *Conference on Genetic and Evolutionary Computation*. 2005. Washington DC.
4. Robic, T. and B. Filipic, *DEMO: Differential Evolution for Multiobjective*. 2005, Jozef Stefan Institute: Slovenia. p. 520-533.
5. Roger, L.S., M.S. Tan, and G.P. Rangaiah, *Global Optimization of Benchmark and Phase Equilibrium Problems Using Differential Evolution*. 2006, National University of Singapore: Singapore.
6. Gamperle, R., S.D. Müller, and P. Koumoutsakos, *A Parameter Study for Differntial Evolution*. 2006, Swiss Federal Institute of Technology Zürich: Zürich.
7. Zoubir, A.M. and D.R. Iskander, *Bootstrap Matlab Toolbox.* Software Reference Manual, 1998.
8. Johnson, R.W., *An introduction to the bootstrap.* Teaching Statistics, 2001. **23**(2): p. 49-54.
9. Zoubir, A.M. and B. Boashash, *The bootstrap and its application in signal processing.* IEEE signal processing magazine, 1998: p. 57-76.
10. *Differential Evolution*. [cited 2006; Available from: http://www.icsi.berkeley.edu/~storn/code.html.

Toward a Unified Framework for Swarm Based Image Analysis

Walther Fledelius[1] and **Brian Mayoh**[2]

Abstract. Swarm based image analysis is a discipline, in which emphasis is on developing swarms with different rules and interactions for obtaining a specific emergent behavior, which may be used in an image analysis context. As the swarm process and results are readily visible, swarm based image analysis is an excellent discipline for obtaining a greater understanding of swarm theory. Currently there is no common way of implementing swarm algorithms for image analysis. This is unfortunate as the emergent behavior is highly sensitive to implementation details, as illustrated here by examples. In order to learn more about the behavior of swarms and share results, such a unified framework is proposed here. Although the focus will be on image analysis, the framework could serve as a basis for a more general swarm algorithm framework.

1 Introduction

Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), are both examples of successful application of swarms to optimization problems. Both have also been adapted and applied to image analysis, see [8] and [7] respectively. Another way of using swarms for image analysis that has been described is, like ACO and PSO, based on the emergence that agents exhibit when they are given simple rules to interact with each other and the environment. But unlike ACO and PSO these algorithms focus on constructing rules and interactions to produce new emergences that can be used for image analysis. The objective was best formulated by Ramos: ”The main goal is to achieve a global perception of one image as the emergent sum of local perceptions of the whole colony.” [9]. In these types of algorithms, the image is the agents digital habitat in which they move around and interact directly with image pixels at their location, as well as other nearby agents. There is currently no common framework for releasing the swarms in the digital habitat as is the case with ACO and PSO for graphs and solution spaces respectively. In the book Swarm Intelligence: From Natural to Artificial Systems, Eric Bonabeau et. al., suggests the creation of a catalog of behaviors that can be achieved by agents interacting by simple rules [2]. It is also noted that this would be an enormous task. But if a unified framework for agents in digital image habitats were established, it would be possible to incrementally build up such a knowledge base. A framework would help share experiences and results from new swarm designs, as well as provide a better understanding of swarm behavior. Another important motivation is that the emergence exhibited by the agents is very sensitive to the actual implementation details, as illustrated here by two examples. Without a unified framework much time and effort would have to go in to describing the swarms in great detail to avoid ambiguous descriptions.

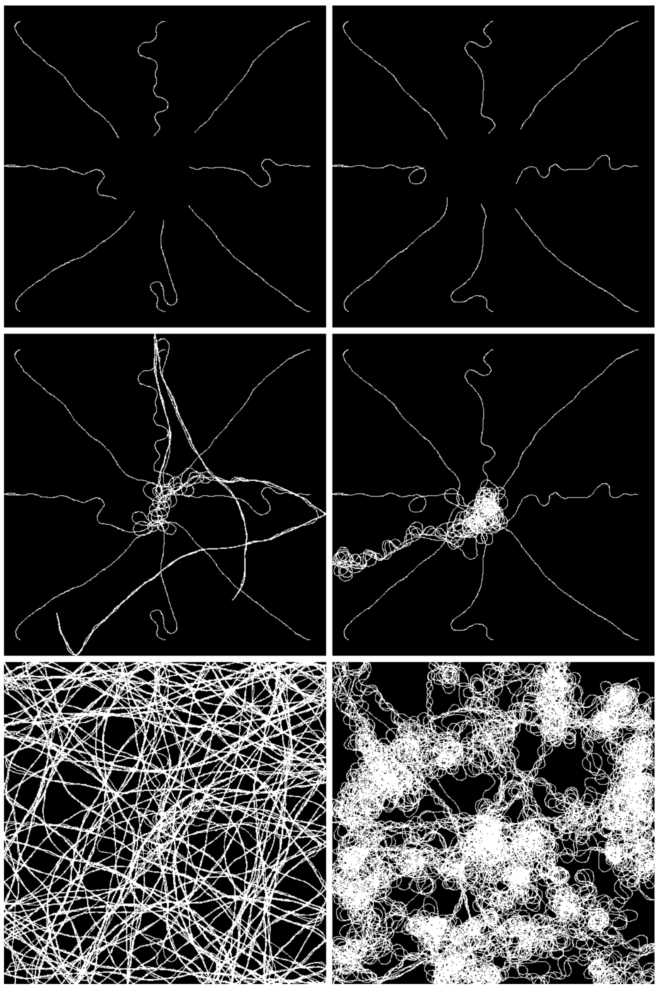


Figure 1. The left and right column show the paths from two different implementations of the same swarm after 100, 200 and 10000 iterations.

2 Swarms in a digital habitat

An important feature of swarm based image analysis, is that an algorithm can be formulated by a set of behavioral rules, without stringent mathematical definitions, thus potentially making it accessible to a

[1] Dept. of Ophthalmology, Aarhus University Hospital, Aarhus Denmark, email: walther@akhphd.au.dk

[2] Computer Science Department, Aarhus University, Aarhus, Denmark, email: brian@daimi.au.dk

larger audience. This method of formulating an algorithm by a rule set can however be deceptively simple, since the emergent behavior is very sensitive to the actual implementation details. The description of an algorithm should besides rules contain detailed information on how internal parameters and motion parameters are updated, as well as when they are updated and in what order. Figure 1 and 2 illustrate how simple swarms descriptions can produce very different emergent behaviors, if details of the implementation is missing or ambiguous.

2.1 Example 1

Figure 1 show two sets of paths, each of eight birds initially placed around the edge of the image, with an initial leftward direction. The birds abide to the following rules:

> ”The birds fly with a random weight of 10% while they move toward the middle of the two nearest birds with a weight of 20%. The speed is constant at 0.5% of the image width”

The only difference between the birds in the left and right column, is that in the right column the birds are all updated before they all move, and in the left column each bird is moved immediately after it is updated. If these birds were used in a search algorithm, this difference in implementation would lead to two very different emergent search strategies. In the left column, two flocks are formed that constantly move without ever remaining in the same region, thus searching a large area sparsely. In the right column, one, two or three flocks are formed, which spend time searching a small area thoroughly. They will suddenly dart off to a new region where they again search that area thoroughly. The difference in the implementations become evident when the distance between agents is close to the their speed i.e. step length. Without the fixed step length the agents positions would collapse in to a single point in space. Agents placed closely around this point, will have preferred directions that are all different from each other, but through almost the same point. When all agents are moved at the same time, they will all move toward the point, and beyond it whereafter their preferred direction changes 180 degrees. It appears as if the agents swarm around a single point. When agents are moved one at a time, the direction of the first agents move, will influence the preferred direction of the next agent, thereby making it possible for the agents to agree on a common direction.

2.2 Example 2

Figure 2 show another example where a flock of 100 birds are used to enhance an image of keratocytes from the human cornea. The keratocytes can be seen as small bright areas in the input image, figure 2A. The birds abide to the following rules:

> ”The birds fly with a random weight of 25%, while they maintain a running average of the pixel values they encounter. The running average is updated with 25%, and the speed is constant at 0.5% of the image width. If a birds running average is higher than the average of the five neighboring birds, the bird deposits nesting material in another image by counting up the pixel value.”

The images B,C and D in figure 2, show the generated image of nesting material by three different implementations of the swarm description (image histograms have been equalized). The difference in implementation between B and C, is that in C the birds leave in a random direction after putting down nesting material, whereas in B they

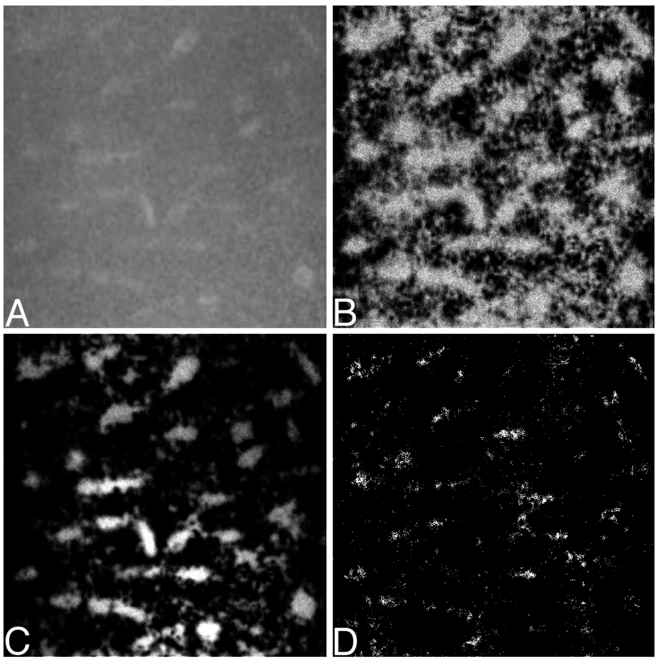


Figure 2. Image A is an image of keratocytes from a human cornea. Images B,C and D show the results of filtering image A with three different implementations of the same swarm description.

continue in the same direction. A bird in the middle of a large bright area, can become trapped if it constantly changes direction, wheres a bird that does not, will always pass through it. The algorithm that changes the birds direction, will therefore deposit relatively more nesting material in the large bright areas. Both interpretations of the algorithm are valid. Biologically there is no reason for a bird to take off in the same direction it landed. However in an actual implementation, one would specifically have to clear the direction and implement a random function in order for the bird to leave in a random direction. It leaves the risk of this information to be assumed implicitly if the algorithm does not change direction of the birds, resulting in an ambiguous algorithm definition. The difference in implementation between B and the D, is that if the birds use the rule of putting down nesting material in D, they are not moved in that iteration. A bird putting down nesting material on a bright pixel will therefore read same pixel again in the next iteration because it did not move. This means it can only leave if the five neighboring birds have an average higher than its own current running average. A great number of birds will therefore remain nesting on the same bright pixel, preventing themselves and others from putting down as much nesting material in other places.

2.3 Reproducibility

The two examples were designed to be as simple as possible, while still illustrating that perhaps seemingly insignificant details of the implementation, can have great influence on the resulting emergent behavior. It is obvious that the results of more complex swarms such as those that have been described in the literature could be very hard to reproduce. Reproducibility is essential in order to learn more about the behavior of swarms, and to share experiences between researchers. A unified framework for deploying agents in digital image habitats will help increase the reproducibility of emergent swarm behavior.

3 Framework

The proposed framework consist of a description of the habitat in which the agents live, as well as a description of the agents. The description of an agent contains information on which type of information an agent can remember, and which states it can assume. Each state, will have a corresponding rule set that affects how agents and habitat are updated. It will also contain a description of an arbiter to handle global decisions and an algorithm structure to bind all the components together in a swarm algorithm. The components are described in the following.

3.1 Digital habitat

The main digital habitat in which the agents live, is the input image to be processed. The agents move around in the habitat, where they can read from, or write to it according to the rule sets. Overlaid the main habitat, there can be a number of equal size parallel habitats, for the agents to interact with. The agents interact with the pixels at their current position, or with the pixels within a predefined action radius from their current position. The parallel habitats can be different color components, different modalities of medical images or the resulting output image. The input data can also be a 3D volume as in [4] where MR scans are processed, only voxels are used instead of pixels. The image can either be an image representing a real world object, or it can be a preprocessed image as in [5], where the pixel values represent gradient directions of the original image. The use of a parallel habitat can be seen in [10] where an additional habitat is used to keep track of areas segmented by the agents, and [1] illustrate the use of an action radius to limit the area where prey pixels are searched. The agents can interact with other agents through stigmergy by changing the habitat as in [9] where agents are depositing pheromones in a parallel habitat, which influences other agents to deposit more in those regions building up a map of the input image. In [3] a colony of spiders uses stigmergy to construct webs for image segmentation.

3.2 Agents

The agents can interpret the habitat as being continuous or discrete. The agent has three motion related parameters, position $\vec{p}$, direction $\vec{d}$ and speed s. In the discrete case, position is typically constrained to pixels, and direction limited to eight values corresponding to vertical, horizontal and diagonal directions. An agent can be in one of a set of states, with each state having its own rule set. Sub populations can be handled by using different states. An agent also has a number of memory cells, where each cell can remember any type of information. Motion parameters, states and memory cells are updated by the rule set. The agents interact with other agents, within a predefined action radius. An agent can obtain information from another agent, and/or change the state of the other agents. In [4] the bloodhound agents communicate to obtain information on direction of track, and in [5] agents can challenge each other whereby the loosing agent changes population affiliation. In [1] a color bank of previously seen pixel values are stored, and in [6] a history of parents are stored. In [10] the scouts can be in one of two states where they respectively do a random search, and label pixels. Similarly [6] uses states to shift between a searching and a reproducing state.

3.3 Rule-set

The agents behaviors are defined by a set of rules. The rule sets are categorized by five types: Initial, internal, state, motion and habitat.

- **Initial rules** are rules that are performed the first time an agent is updated after a state change.
- **Internal rules** update the memory cells of the agent.
- **State rules** can change the state of the agent.
- **Motion rules** can change the position, direction and speed of the agent.
- **Habitat rules** can change the habitat at the agents current position, as well as neighboring agents.

The rules are categorized to ensure an unambiguous order of execution according to the swarm algorithm. If there are more rules of the same type, they should be executed in order of definition. The update the rules perform may be based on habitat, neighboring agents and own parameters.
In the continuous case, the motion rules for changing direction should specify a weight and a unit vector for the desired direction. The speed of the agent is updated like any other memory cell. The agents direction and position is updated in the n'th dimension by a number of rules as:

$$d'_n = d_n(1.0{-}w1{+}w2{+}\ldots{+}wk){+}d1_n w1{+}d2_n w_2{+}\ldots{+}dk_n wk$$

$$p'_n = p_n + d'_n s$$

In previous implementations internal-, state-, motion- and habitat-changes, have not always all been defined by rules, rather by other types of definitions. However by forcing all updates into rules, the risk of ambiguous definitions is minimized.

3.4 Arbiter

The arbiter handles all global calculations and parameters, such as random reproduction, death and pheromone evaporation. Real world agents do not have access to global information and calculations, however it can be necessary in a given real world application. By explicitly assigning the task to an arbiter, global and local information become clearly separated. Pheromones and evaporation is used in [9], and a global centroid of a swarm is calculated and used in [1].

3.5 Algorithm

The algorithm outlines the order in which the swarm and habitat are updated, see pseudo code.
After swarms and habitat have been initialized, all agents are updated once in each iteration, until the termination criteria is met. If an agent has just changed state, the initial rules are performed first, followed by the internal rules. State changing rules are performed until one causes a state change, or there are no more state changing rules. If the agent did not change state, motion and habitat rules are performed on the agent. All agents that did not change state are then moved in their potentially new direction. After all agents have been updated, the arbiter performs global calculations and interactions. Rules of the same type are applied in the order they are described.

```
INIT (habitats and agents)
REPEAT
 FOR (all agents)
  apply initial rules
  apply internal rules
  apply state changing rules
  IF (no state change)
   apply motion rules
   apply habitat rules
  END IF
 END FOR
 FOR (all non state changing agents)
  move
 END FOR
 arbiter: update global and
          population parameters
UNTIL (termination criteria met)
```

4 Framework definition example

The following example illustrate how the bloodhound algorithm of [4] may be described in the proposed framework. Besides the description, there should also be defined values for the constants `c1-9`, as well as detailed information of the functions executed by the algorithm The constants are not transferred from the original article, as the original swarm algorithm definition does not follow the new framework.

```
MEMORY CELLS

 q = Quality of current position
 d = Distance to nearest stationary hound
 t = Iterations remaining stationary
 f = Is in front of nearest stationary hound
 b = Is behind stationary hound

STATE1: MOVING

 Internal rules:
 q = evaluateCurrentPosition()
 d = findDistanceToNearestStationaryHound()
 f = isInFrontOfNearestStationaryHound()

 State changing rules:
 if (q>c1 and d>c2) then State2
 if (q>c1 and d<c3 and f) then State2

 Motion rules:
 alignDirectionWithNearestStationary(c4%)
 moveTowardPointInFrontOfNearestStationary(c5%)
 changeDirectionRandomly(c6%)

STATE2: STATIONARY

 Initial rules:
 assumeDirectionOfImage()
 drawInOutputImage()

 Internal rules:
 b = isAStationaryHoundInFront()
 t = updateTimeRemainingStationary()

 State changing rules:
 if (b) then State1
 if (t>c7) then State1

ARBITER

 Move hounds with more than 15 neighbors to a
 random location

CONSTANTS

 Speed = c8
 Action radius = c9
```

5 Conclusion

A new framework for swarm based image analysis has been proposed, which will help provide a solid foundation for exploring swarms in the future. It provides unambiguous swarm algorithm definitions, which will help share experiences among researchers, and increase the reproducibility of emergent behaviors. The framework is based on a brief review of how swarms have previously been applied, while attempting to encapsulate all the ideas presented in those works.

ACKNOWLEDGEMENTS

The study was supported in part by generous donations from: The Danish Eye Research Foundation and The Danish Association for Prevention of Eye Diseases and Blindness.

REFERENCES

[1] Luis Antón-Canalís, Elena Sánchez-Nielsen, and Mario Hernández-Tejera, 'Swarmtrack: A particle swarm approach to visual tracking', *VISAPP'06, Setúbal, Portugal*, **2**, (2 2006).

[2] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz, *Swarm intelligence: from natural to artificial systems*, Oxford University Press, Inc., New York, NY, USA, 1999.

[3] Christine Bourjotand, Vincent Chevrier, and Vincent Thomas, 'A new swarm mechanism based on social spiders colonies: from web weaving to region detection', *Web Intelligence and Agent System*, **1**(1), 47–64, (1 2003).

[4] Walther Fledelius and Brian H. Mayoh, 'A swarm based approach to medical image analysis', in *AIA'06: Proceedings of the 24th IASTED international conference on Artificial intelligence and applications*, pp. 150–155, Anaheim, CA, USA, (2006). ACTA Press.

[5] Chad George and James Wolfer, 'A swarm intelligence approach to counting stacked symmetric objects', in *AIA'06: Proceedings of the 24th IASTED international conference on Artificial intelligence and applications*, pp. 125–130, Anaheim, CA, USA, (2006). ACTA Press.

[6] Jiming Liu, Yuan Y. Tang, and Y. C. Cao, 'An evolutionary autonomous agents approach to image feature extraction', *IEEE Trans. on Evolutionary Computation*, **1**(2), 141–158, (1997).

[7] Omran M., Salman A., and Engelbrecht A. P., 'Image classification using particle swarm optimization', *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002), Singapore*, 370–374, (2002).

[8] Alice R. Malisia and Hamid R. Tizhoosh, 'Image thresholding using ant colony optimization', in *CRV '06: Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, p. 26, Washington, DC, USA, (2006). IEEE Computer Society.

[9] V. Ramos and F. Almeida, 'Artificial ant colonies in digital image habitats - a mass behaviour effect study on pattern recognition', *Proc. of ANTS'2000 - 2 na Int. Workshop on Ant Algorithms (From Ant Colonies to Artificial Ants)*, 113–116, (2000).

[10] C.E. White, G.A. Tagliarini, and S Narayan, 'An algorithm for swarm-based color image segmentation', in *IEEE SouthEast Conf., Greensboro, North Carolina, USA, IEEE Press*, pp. 84–89, (2004).