Learning and Nonlinear Models (L&NLM) – Journal of the Brazilian Neural Network Society, Vol. 8, Iss. 4, pp. 216-226, 2010.
© Sociedade Brasileira de Redes Neurais (SBRN)

AUTOMATIC NEIGHBORHOOD CONTROL FOR INTELLIGENT AGENTS INSPIRED IN WOLF PACK BEHAVIOR

Luís F. A. Pessoa, Amanda Leonel, Luís C. S. Menezes and Fernando B. de Lima Neto

Polytechnic School of Engineering, University of Pernambuco, Rua Benfica, 455, 50750-470 Madalena, Recife
{lfap, aln, luis, fbln[1]}@ecomp.poli.upe.br

Abstract – This paper aims at presenting an automatic neighborhood control for intelligent agents inspired in the behavior of wolf packs. The outcome of this research is a new tool for modeling the contour of a given cloud of points (that may be represented by mathematical functions) in order to reduce the effort of defining the boundaries of the input points. For that we selected and implemented a small repertoire of wolves like behaviors and some environmental dynamic features. Those features are based on social relations among wolves (the adaptive computing units in the algorithm) and are expressed as two operators, namely, (i) wolf´s location and (ii) wolf´s radius of influence. The put forward algorithm generates a set of location-radius pairs, one for each considered wolf, which is offered *in lieu* of the initial cloud of points. As proof of concept some simulations were carried out and are presented here with clouds of points in shape of four different mathematical functions (two polynomial and two trigonometric). The simulations also aimed at investigating the influence of selected parameters in relation to the observed results. Based on the good results obtained, this preliminary study is thought to be an important step towards an innovative intelligent tool for function approximation based on collective intelligence.

Keywords – Multi-Agent Systems, Natural Computing, Social Computing, Function Approximation, Wolf pack.

Introduction

Function approximation is a very important class of computational problems, as mathematical functions can represent real-world problems or describe clouds of points. Both may involve complicated underlining functions and thus can be computationally expensive. For that, techniques such as Artificial Neural Networks (ANN) and Linear Programming [1] can be applied. If the problem at hand possesses large cardinalities, the former is likely to become expensive in terms of memory occupation and the latter expensive in terms of processing consumption.

As a start point one could consider linear programming using the SIMPLEX [2] algorithm to solve function approximation problems. However, it has an exponential complexity O (2n) - for the worst case. This is undesirable because it means large consumption of computational resources as runtime scales up exponentially with the input size of the approximation problem.

Even though ANN is a fair candidate technique for approximation problems, they can take long periods of time if data preparation and training are also considered. In addition to that, there is also the possibility of overfitting or underfitting when end of training is not appropriately handled. Of course there are many ways to tackle all these difficulties, *e.g.* there are hybrid approaches that use Genetic Algorithms to determine the parameters of the ANN[3]. Even though the existence of all those possibilities to reduce incorrect approximations, they generally result in additional computational costs.

The reasons presented above encouraged us to seek for new intelligent approaches to solve the problem of function approximations. In short, approaches that are able to: (i) model real problems, (ii) implement some reduction in the input cloud of points without loss of representativeness and (iii) perform function approximation in a simple and adaptive manner.

This article aims to use as metaphor the hunting behavior of wolf packs in nature to model cloud of points (representing problem data) and to reduce their cardinality. We believe that the synergy observed when wolves pursue their prey could be transplanted to computer model that are more intuitive and economic in terms of computational resources consumption. This by taking advantage of the interaction between the agents (*i.e.* wolves in our algorithm) and their exerted roles by taking care of several data points, simultaneously.

The environment of the proposed algorithm is composed of: trees – representing the points that belong to the function to be approximated; sheep are the prey for the wolves (*i.e.* points to be "hunted"); and, wolves are predators as they chase the sheep. Following a successful hunt, the wolves are supposed to be strategically positioned so that all sheep will be forced to move towards the outline of the input cloud of points. Hence, the approximated function, in terms of (fewer) points, can be represented by the final positions of wolves and their radii of influence (upon sheep), both obtained during the hunt.

This paper is organized as follows, first of all, the necessary theoretical topics are presented, namely: (i) Wolf pack behavior in nature; and, (ii) Social Computing. Then, the proposed meta-heuristic and implementation is described in detail. Next, obtained results of simulations carried on are commented upon. Finally, conclusions and future perspectives for our approach are presented.

[1] Corresponding author

1 Wolf Pack's Behavior in Nature

Wolf packs are usually made-up by groups between 8 to 15 wolves. This, of course, depends on available hunting area, food stock and other intrinsic social factors. The wolf packs in nature also have a well-defined hierarchy, determined by wolves' posture, behavioral characteristics and struggle for obtaining pack dominance.

The dominant wolf couple is called alpha. They are the first to feed and the only ones that reproduce in the pack. Below, in the hierarchy, there are beta wolves exercising leadership just below the alpha couple. Besides that, at the bottom of the pyramid, there are omega wolves that have almost no rights within the pack.

In opposition to what many people think, wolves do not communicate only through howling but also through their body posture and body odors [4]. Their many ways to communicate allow them to delimit territory, to maintain cohesion of the pack in dense environments like forests, to gather the pack at specific locations or even, to express fear, submission, dominance, etc.

Hunting is carried out as a group task in which all members of pack take participation. Their strategy is to chase the prey until they become tired, by means of surround them. In general, they aim the oldest, youngest or injured preys. By doing so, the group spends less energy in the pursuit. Once the prey is killed, wolves eat quickly to prevent theft or dispute for food with other animals.

Surrounding the target and operating in coordination to achieve common goals are the key aspects to be incorporated into the artificial metaphor put forward in this paper.

2 Social Computation

Social Computation is a subarea of Multi-Agent Systems with a focus on cooperation and collaboration among agents that compose the environment through interaction, communication and collective behavior. Thus, two interesting paradigm advantages for the problem at hand are: (i) synergistic behavior and (ii) burden distribution between pack members. Both features result in efficiency and efficacy.

Social Computation can also be seen as a sub-area of Natural Computing. It is Central to Social Computation the notion of social group, which is a set of agents with collective behavior (*i.e.* a crowd, a herd, a flock, etc.). Although agent movement in the environment is not mandatory, the existence of communication or some interaction between them is seminal. In short, this construction affords:

- Simulations of the dynamics of real systems and analyzes of specific interaction aspects between agents that make-up the environment. It is therefore possible for one to forecast impact in the group or in the environment of changes produced by the emergent interactive dynamics; and,
- Solutions of complex problems with low computational costs and in a more intuitive manner [5].

Despite modeling and implementation in social computation generally been made using Multi-Agent Systems, various applications also incorporate some mechanism for Evolutionary Computation [6] in order to improve the adaptability of agents within the population.

The idea of '*social group*' is used here to refer to a set of agents that can interact among themselves aiming at a common goal (*i.e.* function approximation). In the proposed model, cooperation among agents was carefully planned so that the computational cost is kept low, thus, providing a more realistic model in relation to the metaphor of packs in the nature (*i.e.* goals and intentions do not conflict).

3 Modeling and Implementation of the automatic neighborhood control

The proposed algorithm was inspired by the hunting behavior of wolf packs in nature. In the new algorithm, we devised that during the hunt, wolves will chase their prey (by adjusting their positions) and adjust their radii of influence upon preys, which determines the set of sheep that are being hunted at a specific time. The trees represent the input cloud of points which one wants to model.

In a reactive manner, sheep run towards the forest every time they lie within the wolf's action radius. Thus, at the end of a processing cycle of execution, all wolves will be positioned at strategic locations with their action radii adjusted as such to command one or more sheep that are likely to be positioned on the boundary of the forest (*i.e.* the cloud of points). Figure 1 shows a schematic view of the proposed model and the elements that compose it.

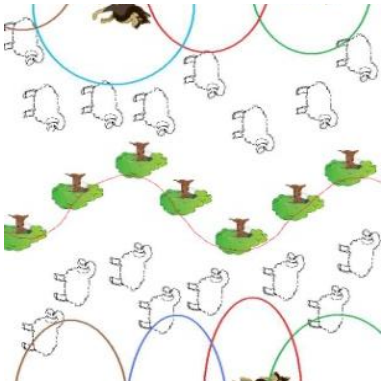


Figure 1 – Schematic view of the conceived meta-heuristic.

Figure 1 also shows that there are actually two packs chasing the sheep, one at each side of the forest. This design decision was taking due to better tackle the concavities turned up and down when the algorithm is approximating some convoluted data such as in some mathematical functions.

It should be noted, however, that no explicit communication protocol between the wolves was included so far. Nevertheless, as will be discussed later, the wolves can communicate indirectly through their actions and through perceived sheep's reaction. It is noteworthy that a small number of wolves is generally able of pursuing the tasks presented in section 4.

3.1 Description of the Elements of the Environment

The wolves are goal based agents whose intentions are: (i) to decrease their individual mean errors and (ii) to decrease their individual standard deviations based on their mean errors. The mentioned error will be formalized later but can be seen as the sum of mean sheep distances to the forest, for all sheep that lay inside the action radius of a given wolf. To achieve their goals, the wolves can modify the environment through their operators by increasing their action radii and re-positioning themselves. Both operators will be detailed later.

As mentioned before, the sheep are simple reactive agents. Thus, they react to every wolf's action every time they enter a wolf action radius. The sheep's action will always be to move out of reach of wolves and as effortless as possible towards finding shelter in the forest direction. In other words, they go to the outline of wolves' action radii. This happens because as soon they reach the outline of the action radius they no longer became subject to any threat of a particular wolf.

The trees, the last element of the environment, are static. They are positioned when the algorithm starts, representing the cloud of points in a one-to-one relation or a one-to-few (depending on preprocessing). Their goal is solely to guide the computation. That is, wolves will try to position themselves in a way that all subject sheep find shelter near the closest tree.

3.2 Computational Model

Figure 2 shows how the conceptual model explained before was implemented as an algorithm. Notice the wolves (center of circumferences) with their respective action radii (circumferences); the sheep on the boundary of the action radii (asterisks); and, the forest (in the middle of the image), which represents the input cloud of points (small crosses).

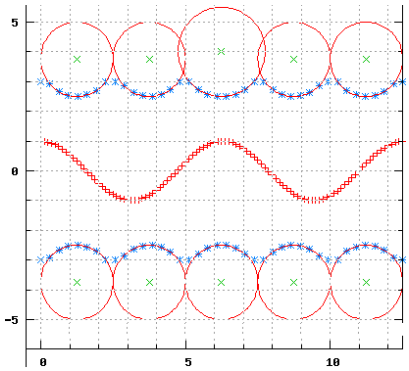


Figure 2 – Computational environment with all elements of the proposed model.

In this first implementation, the cardinalities of wolves and sheep are constants along all period of "hunting" and defined by two specific parameters. The number of trees is assumed equal to the number of points in the input cloud (of points). Besides that, we assumed that for each wolf at one side of the forest, there is an equivalent one at the same abscissa of the Cartesian 2D plan. The same happens with the sheep, according to the selected set up.

The implemented environment has other restrictions that reflect the conceptual model: (i) the sheep cannot trespass the forest limit because they would be susceptible to the opposite side wolves; and, (ii) the wolves is not overcome the sheep, considering that it is an unlikely situation during the hunt.

3.3 System Dynamics

The execution of the proposed algorithm iterates over all wolves of the pack (as if they are in a circular list). Each selected wolf assesses the current environment state and possible actions. When the last wolf of the list deliberates about its actions, the process restarts and the first wolf is selected again. Each cycle determines an epoch of processing.

Each wolf deliberates about its next action, based on its mean error (E_L) and on its standard deviation (σ_E). The wolf mean error is the sum of all individual sheep (O) mean distance to the forest ($d_{O,F}$) – that lay inside the action radius of the particular wolf. Equations below defines wolf error (1) in terms of distance from sheep to forest (2); equation (3) defines the standard deviation of a particular wolf.

(1)

where, n is the number of sheep that are inside the wolf action radius, F is the correspondent point of the forest.

(2)

Sheep to forest distances ($d_{O,A}$) are calculated as the Euclidean between the sheep and the tree (A) closest to the abscissa of that sheep. Once the wolf has computed its individual mean error, its standard deviation is then calculated according to (3).

(3)

Figure 3 shows how the calculation is carried out for obtaining the distance between three sheep (A, B, C) and their respective trees (*i.e.* trees closer to the sheep abscissas).

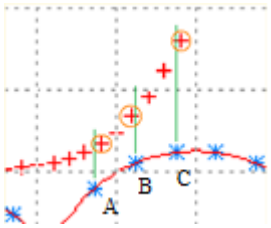


Figure 3 – Distance between sheep (asterisks) and trees of the forest (crosses).

Once the mean error and standard deviation values are calculated, the wolf selects which action to take, that is, wich operator will be used: (i) *position adjustment* – in case mean error and standard deviation are high; or (ii) *radius adjustment* – in case mean error is high and standard deviation is low. High and low are defined in accordance to specific parameters. For example, if the standard deviation of the wolf is smaller than the wolf's minimum standard deviation parameter, this means that the standard deviation is low. Adjustments of radius and position are shown in Figure 4, (a) and (b), respectively.

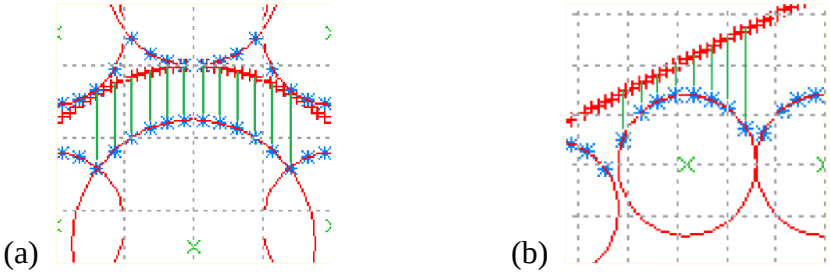


Figure 4 - Examples of radius adjustment (a) and positioning adjustment (b).

After each wolf is dealt with, the system will check if a stop criterion is met. If so, execution stops; if not, another wolf will deliberate about its actions as explained before. The system stop criteria are: maximum number of epochs and overall minimal global mean error (*i.e.* the sum of mean sheep distances to the forest). The flowchart given in Figure 5 shows the computational steps explained so far.

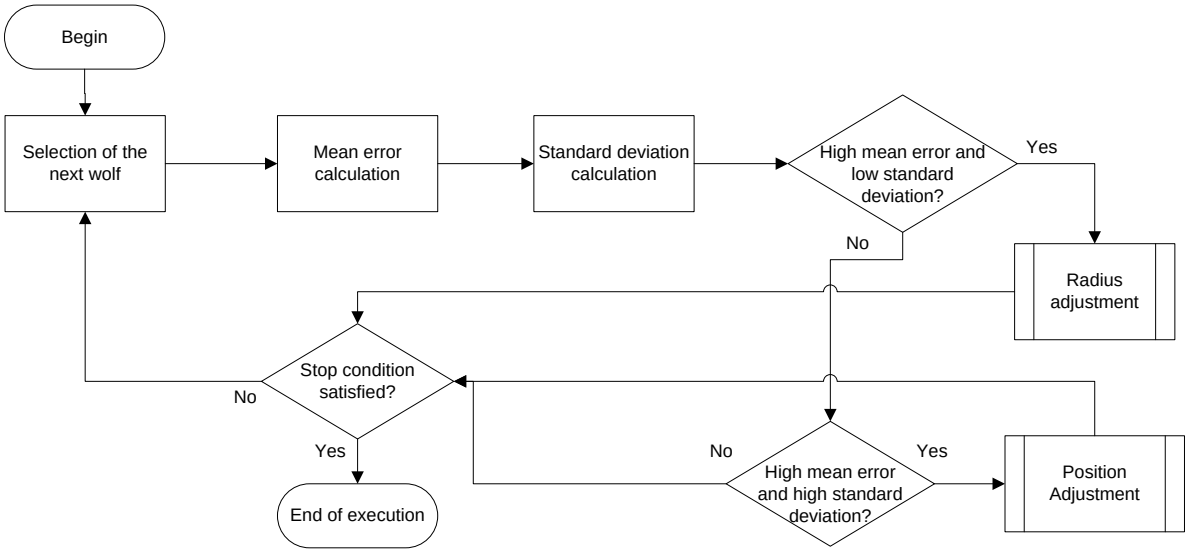


Figure 5 – Algorithm main steps.

It is important to emphasize that a wolf will not adjust its radius or positioning using pre-determined values. This is a security measure as the sheep could overshoot the forest. Thus, new values for radius or position are 90% of the distance of the shortest sheep included within wolf's action radius from the forest. Figure 6 shows examples that illustrate that.

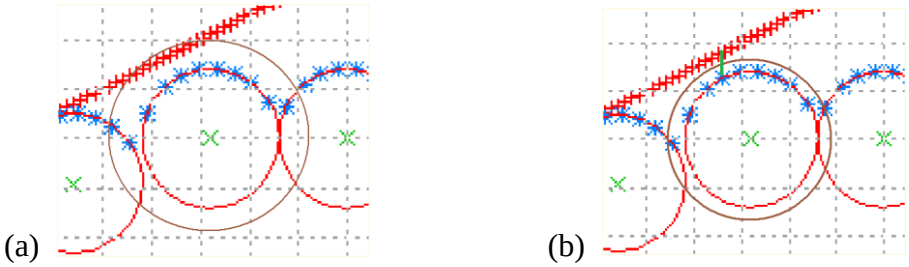


Figure 6 - Prohibited radius adjustment (a) and a valid adjustment (b).

At the end of the algorithm execution, the final output will be the set of wolves' positions (*e.g.* Cartesian coordinates) and their respective action radii. These values are then considered *in lieu* of the given cloud of points. Notice that, the wolves-trees ratio gives the compression rate of the function approximation. That is, more wolves would, for example, increase the quality of the representation; conversely fewer ones would produce a more compact model for the input data.

4 Experiments

This section presents the results of all experiments carried out to analyze the operation of the put forward algorithm. Here, we discuss the method, variables and parameters used in the experiments. To illustrate the flexibility of the algorithm different configurations of trees (*i.e.* forests) were selected, namely: a linear function, a quadratic function, the cosine and the hyperbolic tangent functions.

4.1 System Parameters

The devised parameters of the computational model can be classified into four classes: initialization, functional, stop criteria and environmental parameters.

The parameters that are readily obtained directly from the environment are: number of environment dimensions and the input file that represents the forest (*i.e.* cloud of point).The parameter that are related to the approximation functionality of the algorithm are: number of trees (*i.e.* the cardinality of the cloud of point), initial number of wolves and initial number of sheep. The parameters that define how the algorithm will be initialized are: the wolf's inner function (kept constant here as Gaussian although it could be other), wolf's minimum mean error, the initial distance of wolves to the sheep, the initial distance of the sheep to trees and wolf's minimum standard deviation. The two stop criteria are: epochs number and minimum error.

Table 1 shows all the system parameters. Those that are marked as *Variable* were chosen to be analyzed during the experiments presented in this paper. The other were set based on previous exploratory simulations, but can also be varied.

Table 1 – System parameters per classes (E – Environmental, F – Functional, I – Initialization, S – Stop Criteria).

Parameter	Class	Value	Parameter	Class	Value
Dimensions Number	*E*	*2*	**Wolf's Minimum Error (MME)**	*I*	*Variable*
Input File	*E*	*Variable*	**Initial Distance from Wolf to the Sheep**	*I*	*1 Cartesian unit*
Trees number	*E*	*100 trees*	**Initial Distance from Sheep to Tree**	*I*	*2 Cartesian units*
Wolves Number (WN)	*F*	*Variable*	**Wolf's Standard Deviation (MSD)**	*I*	*Variable*
Sheep Number	*F*	*40% of the trees*	**Epochs Number**	*S*	*60*
Wolves' inner Function	*I*	*Gaussian*	**Global Minimum Error**	*S*	*0.05 Cartesian units*

4.2 Experimental Method Used

The experiments were designed using 2^k Factorial design [7] with three factors (k = 3), resulting in eight simulations for each function to be approximated. This to understand and analyze the effects of each studied factor on the variable responses. The main objective though was to determine whether the system works as initially conceived, that is, if the sheep will outline of the forest.

The factors studied in the four experiments (*i.e.* four functions) were: the number of wolves, the wolf's minimum mean error (MME) and the wolf's minimum standard deviation (MSD). As mentioned before, MME and MSD are the thresholds that define if the wolf individual error or its standard deviation, respectively, is high or low. The levels for each studied factor were: initial number of wolves (5% and 20% of the trees number), MME (0.5 and 0.15 Cartesian units) and MSD (0.5 and 0.15 Cartesian units).

The three factors – MME, MSD and the initial number of wolves – were chosen because their direct influence in the final results. Moreover, their variations resulted in marked changes in system performance. On the other hand, each factor's pair of values was chosen arbitrarily so that they varied between two extremes in order to allow a more conclusive analysis of their impact onto the system.

For example, within an environment with the forest cardinality equals to one hundred the average rate between sheep and wolves varies between 20:1 and 5:1. Thus, it is possible to analyze whether the wolves have more success to "hunt" a higher or lower number of sheep, this combined with the two other factors' variations: MME and MSD.

Four different configurations were chosen for the input cloud of points. Two of them representing polynomial type of functions (*i.e.* linear and quadratic functions) and the other two, trigonometric type of functions (*i.e.* cosine and hyperbolic tangent functions). These functions were chosen so that the algorithm could be tested for different conformation of forests. Obviously, in a real world application the cloud of points could not be easily defined by a function, hence another possible use of our algorithm.

4.3 Linear Function Approximation

The selected linear function is given by (4).

(4)

The most important factor among those analyzed was the number of initial wolves, followed by the MME. For the worst case, the global mean error obtained was 0.47177, and standard deviation of 0.45831. It was observed that when MME assumed the lowest value, it took more time for the algorithm execution to converge than any other tested configuration.

The best result of the experiments produced an overall average error equal to 0.09772, with standard deviation of 0.05912. The best and worst results were reached using the following parameters, respectively: MME = 0.15, MSD = 0.15 and

WN = 5% of the forest; and, MME = 0.5, MSD = 0.15 and WN = 5% of the forest. The final states for the best and worst case are shown in Figure 7, while the graphs of global mean error decay for the worst and best cases are presented in Figure 8.

It can be seen in Figure 7 that a small number of sheep causes a higher standard deviation. So that, most sheep within the wolf's action radius are far away from the cloud of points. Nonetheless, the result of this experiment is not completely satisfactory, because the sheep cannot form a perfect outline of the forest and thus have low accuracy, the algorithm is still able to make an approximation.

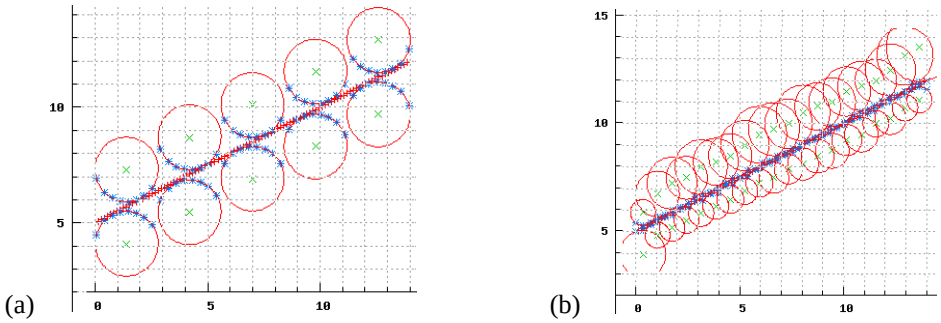


Figure 7 - Worst (a) and best (b) results for linear function.

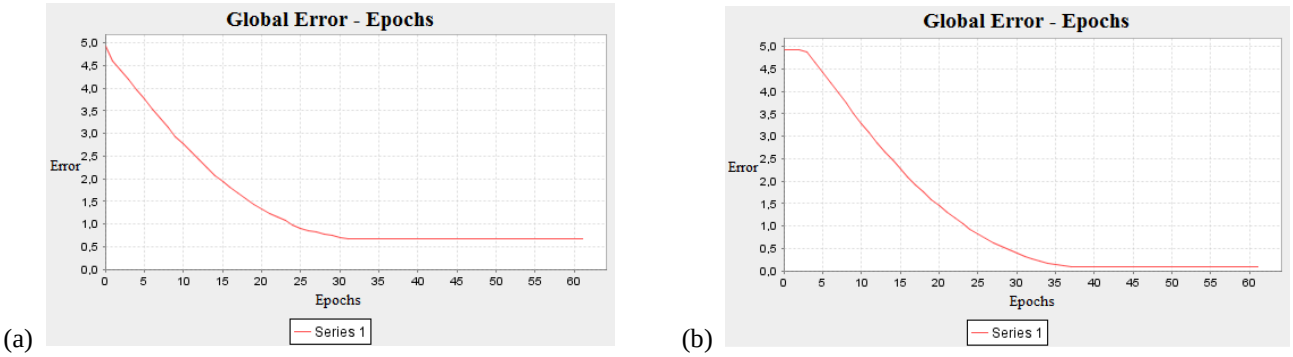


Figure 8 - Global mean error decay for the worst (a) and best (b) results.

When the mean wolf's standard deviation is quite small - the sheep are very close to the forest. Note also that despite the use of quadruple number of wolves (in comparison to the worst case), the number of epochs required for the algorithm convergence increased by only 1/6.

4.4 Quadratic Function Approximation

The forest described by the quadratic function is given by (5)

(5)

For this so called given cloud of points, the most important factor was the initial number of wolves, followed by MSD. The worst case obtained a global mean error equal to 0.47849 and a standard deviation of 0.40943. On the other hand for the best case, the global mean error was equal to 0.11890, and the standard deviation was equal to 0.09962.

The best and worst results for the quadratic function were reached using the following parameters, respectively: MME = 0.15, MSD = 0.15 and WN = 5% of the forest; and, MME = 0.5, MSD = 0.15 and WN = 5% of the forest. Notice that it was the same configuration for the linear function. Figure 9 shows the final stages for the best and worst cases when the forest is the quadratic function, while Figure 10 shows the graphs of the global mean error decay for the same worst and best results.

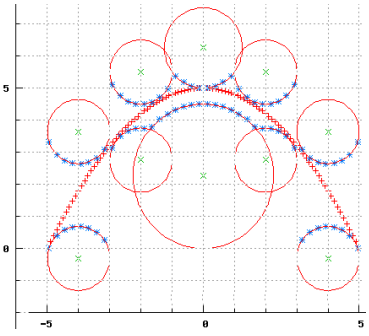
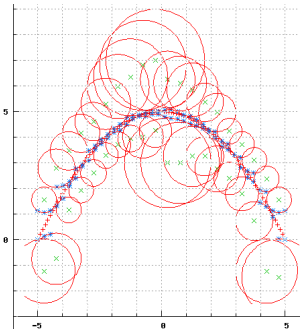
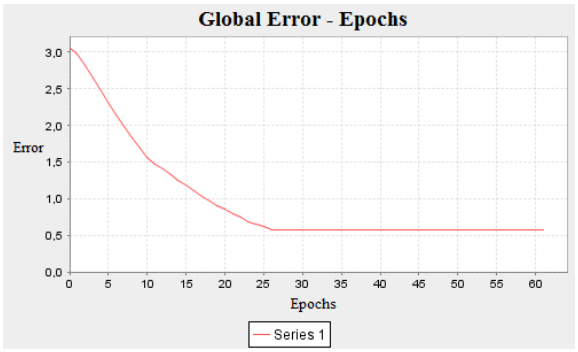
(a)  (b) 

Figure 9 – Worst (a) and best (b) results for the quadratic function.

(a) 

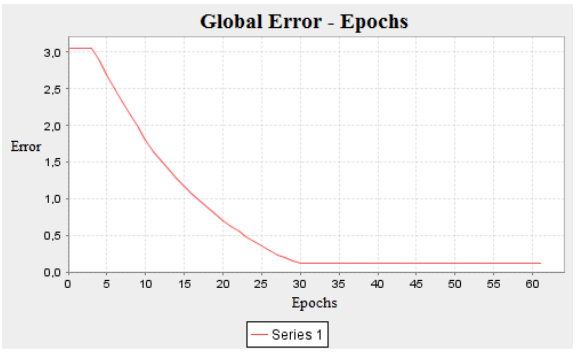
(b) 

Figure 10 - Global mean error decay for the worst (a) and best (b) results.

The worst result for this type of function revealed an interesting behavior, namely, the wolf centered on the parable concavity worked precisely as planned. That is, the best strategy to hunt the sheep was to increase its radius of action. In this way many sheep will be closer to the forest commanded by a single wolf. In this particular case the other wolves did not increase further theirs range of action. They rather moved due to the inner function selected and other parameter.

The global mean error for the quadratic function was very similar to the corresponding graphs for the linear function. The main difference is the faster convergence of the algorithm for the quadratic function. But both spent more time to converge when the number of wolves in the environment quadrupled.

4.5 Cosine Function Approximation

The cosine function used was defined within the interval [0, 12.5]. Thus, it was possible to obtain some concavities facing up and others facing down.

The factor that resulted in a larger variation of the final results was, as well, the initial number of wolves, followed by MSD. The worst results for this function were global mean error of 0.47950 and a standard deviation of 0.4291. On the other hand, the configuration that obtained the best result produced the global mean error was 0.11100, and the standard deviation was 0.07625.

Figure 11 shows the final states for the worst and the best result obtained to this forest configuration, while Figure 12 shows the corresponding graphs of the global mean error decay for each case. For the cosine functions, the best and worst results were reached using the following parameters, respectively: MME = 0.15, MSD = 0.15 and WN = 5% of the forest; and, MME = 0.5, MSD = 0.15 and WN = 5% of the forest.

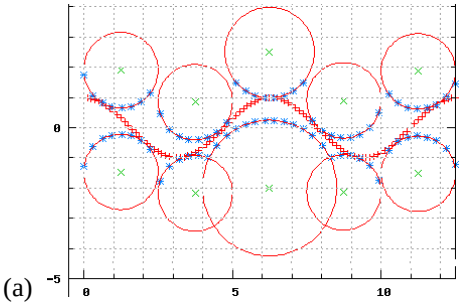
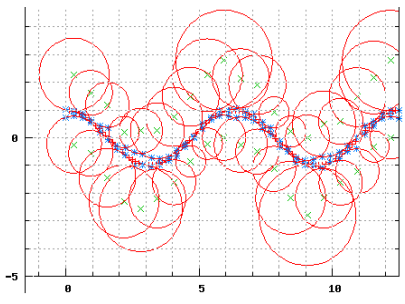
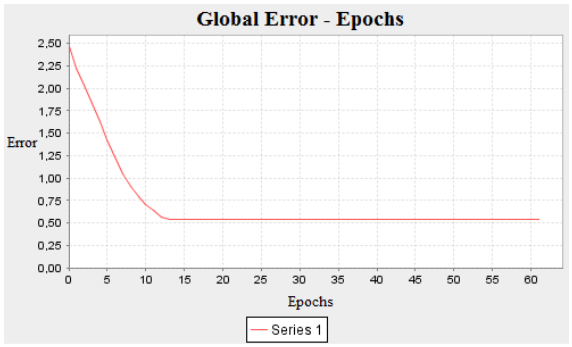
(a)  (b) 

Figure 11 - Worst (a) and best (b) results for cosine function.

(a) 

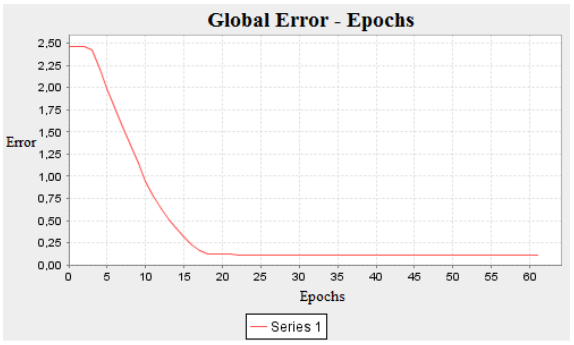
(b) 

Figure 12 - Global mean error decay for the worst (a) and best (b) results.

The results obtained with the cosine function were very similar to those obtained with the quadratic function. However, we found significant improvement in the cosine function, because in all experiments with this function the standard deviation was lower than the overall average error. Therefore, the distances of the sheep to the trees did not vary as much as in the previous experiments. Another relevant result for this experiment was the fast convergence of the algorithm, if compared with the previous two experiments.

4.6 Hyperbolic Tangent Function Approximation

Equation (6) was used to represent the forest in this experiment.

(6)

Similarly to the linear function experiment, the factor of greatest importance here was the initial number of wolves, followed by the MME.

The worst result produced a global mean error of 0.52966 and a standard deviation of 0.58227. On the other hand, the best case resulted in global mean error of 0.13242 and a standard deviation of 0.08611. Figure 13 shows the final states for the worst and the best result and Figure 14 shows the respective graphs of global mean error decay. The best and worst results for the Hyperbolic Tangent function were reached using the following parameters, respectively: MME = 0.15, MSD = 0.15 and WN = 5% of the forest; and, MME = 0.5, MSD = 0.15 and WN = 5% of the forest. The variables values for both cases were the same of all other functions tested above.

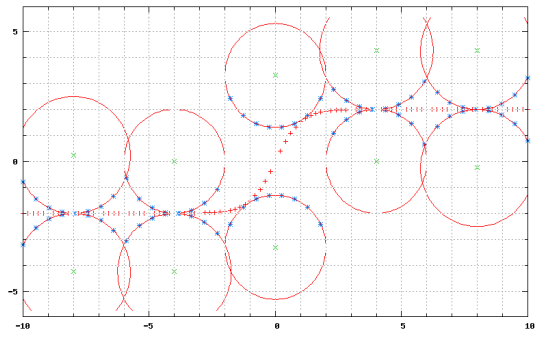
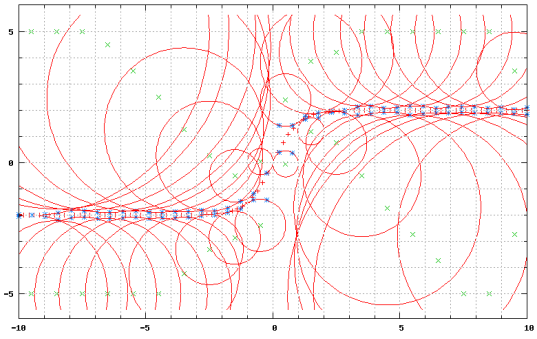
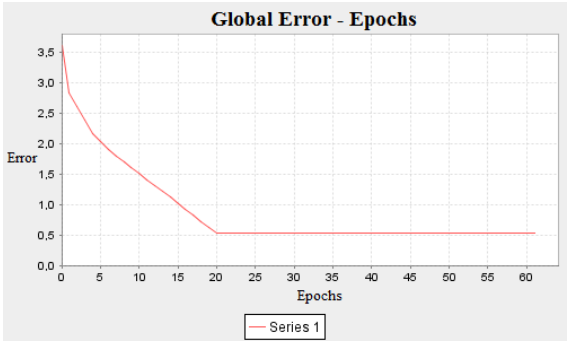
(a) 
(b) 

Figure 13 – Worst (a) and Best (b) results for the hyperbolic tangent function.

(a) 

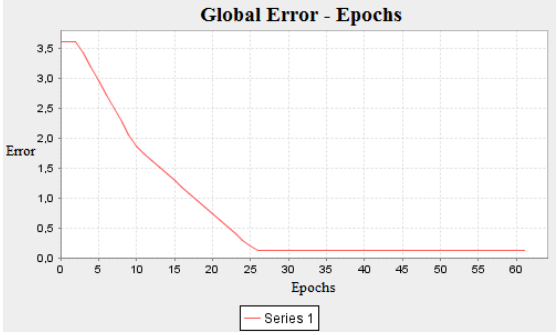
(b) 

Figure 14 – Global mean error decay for the worst (a) and best (b) results.

It is noteworthy that for the worst case, the wolves have not hunt the sheep properly so they formed an ill outline of the hyperbolic tangent function. While a few sheep are very close to the forest, others stayed far away, because the high global mean error and standard deviation.

As in almost of the simulations for that function, it was possible to notice that the wolves became positioned symmetrically across the Cartesian plane; and this was not programmed. It emerged starting from the interaction among elements of the environment and also due to the symmetry of the hyperbolic tangent function. This feature is very interesting because, if indeed wolves' behavior obey the symmetry of a given cloud points, it will only be necessary half of the wolves to represent the whole input. The other half could be obtained as a projection to the other side of the cloud of points. This means that a small number of wolves could satisfactorily model such cloud of points.

Another curious behavior that could be observed in the final states of the execution of the algorithm (mainly in the best result for this function) was that some wolves became orthogonally aligned in the direction where changes on the hyperbolic tangent inflexions happen. This is another type of emergent behavior that was not planned but was observed. Although, interesting more simulations are still necessary to assess the prevalence of such type of behavior. With regard to the decay of the global average error, the pattern remained similar to previous experiments.

5 Conclusions

This paper presents an innovative approach for modeling the contour of a cloud of points. We aimed at reducing the representativeness of that given points through a new approach based on the hunt behavior of wolf-packs. In the model proposed the wolves hunt the sheep by their position and radius adjustment operators. So, they force the prey to run towards the forest (*i.e.* the given set of points). At the end of algorithm execution, the cloud of points can be represented by the set of final position of each wolf and their respective radii of influence. Next, we carried out experiments that aim to validate the conceptual model and identify the relevance of some parameters (*e.g.* compression ratio) to obtain a better final result.

The first general observation is that for all functions examined, the worst case obtained (among the eight configurations of the factors tested) was number of wolves = 5% of the trees; MME = 0.5; and MSD = 0.15. On the other hand, the best cases were obtained for the following configuration: number of wolves = 20% of the trees, MME = 0.15, and, MSD = 0.15.

It is noteworthy that the results obtained with the present algorithm implementation, for the same parameter configuration and the same forest, were always identical. This happens because the wolves are deterministically positioned in the environment, which easily could not be the case for diversity sake.

From the results obtained, we also conclude that the number of wolves in the environment is essential to achieve a better outline of the forest. Considering that, when the configurations had a smaller number of wolves, the sheep did not adhered well to the forest. Also, it was noticed that MME and MSD have the same high importance in the system. Their values variation greatly influences the final result (*i.e.* global mean error), especially if combined with an increase in the number of wolves in the environment (*i.e.* wolf pack).

So far we observed that the number of wolves should not be low (<5% of trees) or very high (> 70% of trees). A low number of wolves may cause an increase of global mean error. This means that wolves final positions and their final radii are not properly suited to the forest. At the other edge of the scale, if the number of wolves is larger than necessary, the ratio between the cardinality of the wolf pack and the forest will be close to one, indicating that there is approximately one wolf for each tree. Thus, it is not worthy the use of the algorithm because in this way there will be no reduction in the cardinality regarding a given cloud of points.

In the current version of the algorithm, the sheep just react to the wolf's actions. They serve to guide the wolf pack actions and help to define the final position and radius for each wolf. Its initial population is also essential to a better functioning of the algorithm. For example, assuming a forest of cardinality 100 and a sheep population of 20, it is possible to reach a global mean error close to zero. However, this does not mean that the final approximation will be good as the representative of the majority of points is to be lost due to the small amount of sheep. On the other hand, if we assume an initial number of sheep in the environment equal to 90, there is a considerable increase of the computational cost and there is no guarantee that the result will be much better when considering a population of 20 sheep. Therefore the initial population of sheep has a great influence on the accuracy of modeling the function outline.

Even though there is not an explicit communication protocol between the wolves, there is an indirect communication between them. For their actions are based on sheep within their action radius and, as there are intersections between the wolves' radii, a wolf's action may interfere with the action of his neighbor. Figure 15 illustrates an example of this interference. In case of the selected wolf uses one of its operators, the highlighted sheep will run towards the trees. Thus, not intentionally the centered wolf reduced the mean error and the standard deviation of the others next to it, maybe interfering in next decisions of closer wolves.

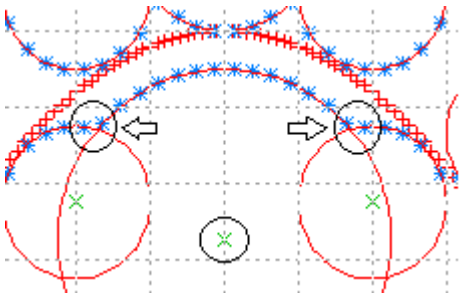


Figure 15 – Indirect communication between the wolves.

Although presented here as a proof of concept, the proposed algorithm represents an candidate tool for modeling the outline of a given cloud of points. We highlight the high degree of autonomy it possesses even without other operators such as wolf recruitment as commented before. In the long run, it well be that this algorithm could be used to solve more complex problems of function approximation, hopefully, keeping a low consumption of computational resources.

6 Future Works

To make the algorithm less dependent on the initial number of wolves, future studies should investigate a new way (operator) to deal with wolf recruitment and disposal. Thus, the number of wolves may be adjusted during the algorithm execution to better model the problem, that is, to represent the given data set (*i.e.* forest).

Another aspect to be investigated is how to alter the ratios between sheep and wolves, and trees and data points (in order to save computational resources).

In addition to that, to make the algorithm less user dependent and also more robust one should consider including intelligent mechanisms for defining other parameters values according to problem features. Furthermore, the development of a protocol and a mechanism of communication between the wolves would enable better coordination of their actions. Other improvements to be made in the algorithm are: perform tests on the algorithm scalability and determine when a wolf pack on each side of the forest is really necessary.

7 References

[1] Jayadeva, A. K. Deb, S. Chandra, Algorithm for building a neural network for function approximation, **IEE Proceedings – Circuits, Devices and Systems**, 149 (2003), 301 – 307.

[2] J. Dobgenski, Programação Linear para Aproximação de Funções Aplicada ao Projeto de Filtros Digitais, Master Thesis in Electrical Engineering (in Portuguese), **UNICAMP**, (1997).

[3] G. A. Rovithakis, I. Chalkiadakis, M. E. Zervakis, High-order neural network structure selection for function approximation applications using genetic algorithms, **Systems, Man, and Cybernetics, Part B: Cybernetics**, 34(2004), 150 – 158.

[4] Wolf Country, The Wolf Pack, Available at: <http://www.wolfcountry.net/information/WolfPack.html> Accessed on 2009 Sep 21.

[5] E. Bonabeau, M. Dorigo, T. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, **Oxford University Press**, (1999).

[6] F. B. Lima Neto, M. R. S. Pita, H. S. Barbosa Filho, Hybrid and Evolutionary Agent-Based Social Simulations Using the PAX Framework, **In: Nineth International Conference on Intelligent Systems Design and Applications - ISDA**, (2009), 497 – 504.

[7] Averill Law, Simulation Modeling and Analysis, 4ª Edition, **McGraw-Hill**, (2006).