6

Simulation Optimization Using Swarm Intelligence as Tool for Cooperation Strategy Design in 3D Predator-Prey Game

Emiliano G. Castro and Marcos S. G. Tsuzuki
Polytechnic School of University of São Paulo
Brazil

1. Introduction

It has long been established that simulation is a powerful tool for aiding decision making. This is due to a number of factor, such as the inherent ability to evaluate complex systems with large number of variables and interactions for which there is no analytical solution. It is considered as a tool to answer "what if" questions, not a solution generator technique. This scenario could be changed with the aid of optimization procedures, and so simulation could answer not only "what if" questions but also answers "how to" questions, providing the best set of input variables and maximize or minimize some performance measure. For the purposes of optimization, the simulation is simply a "black box" which computes a realization of the (stochastic) function value for a given combination of the parameter values (Magoulas et al., 2002).

Simulation optimization is the process of finding the best values of some decision variables for a system where the performance is evaluated based on the output of a simulation model of this system. The components of a typical optimization system are present in a simulation optimization system: decision variables, objective function and constraints. The decision variables must be contained in some feasible region defined by the constraints. The objective function is a real valued function defined on these variables, but due to the complexity and stochastic nature of the underlying system an analytical expression can not be determined and the value returned by the stochastic simulation must be used instead.

The predator-prey game is a multi-agent system, which originally came from the field of Artificial Intelligence (AI). At first, this field was called Distributed Artificial Intelligence (DAI); instead of reproducing the knowledge and reasoning of one intelligent agent as in AI, the objective became to reproduce the knowledge and reasoning of several heterogeneous agents that need to jointly solve planning problems. Some researchers have focused more on the agent and its autonomy (for instance, the definition of an agent proposed by Wooldridge and Jennings (1999): "an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives"), while others, engaged in the field of multi-agent systems, have focused more on the organization of multiple agent interactions (Huhns & Stephens, 1999).

The aiming of this work is to propose a simulation optimization tool for solutions' synthesis to a problem involving a dynamic and complex system (to bring forth strategies for a

Source: Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Book edited by: Felix T. S. Chan and Manoj Kumar Tiwari, ISBN 978-3-902613-09-7, pp. 532, December 2007, Itech Education and Publishing, Vienna, Austria

hunting game between predators and prey in a three-dimensional environment) whose solution involves their agents' emergent behavior (in this case, the predators'). Moreover, this simulation optimization uses the PSO as optimization algorithm which in turn, is also based on emergent behavior. That is, a tool that employs emergence to project emergence.
The organization of this work is as follows: Section 2 contains a review of the Particle Swarm Optimization (PSO). Section 3 describes the 3D predator-prey game: the predator and prey behaviors, the parameter of the hunting game and predator's model of strategy. Section 4 presents the developed tool. The results are presented in Section 5, and some concluding remarks are given in Section 6.

2. Particle Swarm Optimization (PSO)

PSO is based on the collective motion of a flock of particles: the particle swarm. Each member of the particle swarm is moved through a problem space by two elastic forces. One attracts it with random magnitude towards the best location so far encountered by any member of the swarm. The position and velocity of each particle are updated at each time step (possibly with the maximum velocity being bounded to maintain stability) until the swarm as a whole converges.
The update rule for PSO contains only two parameters: the relative importance of the influences on a particle of the particle best and the swarm best solutions, and the number of particles to be used as neighbor. A particle can use as reference the best result obtained by any particle of the swarm, or just for their neighbor particles. In that case, the analogy is in the topological mean, and not for the eventual momentary proximity in the parameter space. Thus, the size of this neighborhood to be considered is a variable in the algorithm.
Perhaps inspired by the original derivation of PSO (an abstract version of the factors involved in the feeding behavior of flocks of birds), early progress in PSO often took the form of adding terms based on biological or physical analogies. One of the most successful of these was the "inertia weight" (a friction coefficient added to the velocity update rule). It is employed to control the impact of the previous history of velocities on the current velocity. In this way, the "inertia weight" regulates the trade-off between the global and local exploration of the swarm. A large inertia weight facilitates global exploration, while a small one tends to facilitate local exploration, fine-tuning the current search space.
The PSO happens in accordance to the following formula:

$$\mathbf{v}_i(t)=w\,\mathbf{v}_i(t-1) + \varphi_1\,[\mathbf{p}_i - \mathbf{x}_i(t-1)] + \varphi_2\,[\mathbf{p}_g - \mathbf{x}_i(t-1)] \quad (1)$$

Where $\mathbf{p}_i$ is the best position found by the particle i so far, $\mathbf{p}_g$ is the best position among all particles, and φ_1 and φ_2 are positive random variables, evenly distributed in the intervals [0, φ_{1max}] and [0, φ_{2max}], calculated at each iteration for each particle. w is the inertia weight. The position of each particle is updated at every iteration using the vector of velocity (adopting the time unit as the iteration step):

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t) \quad (2)$$

PSO belongs to the class of the evolutionary algorithms based on population, as well as the genetic algorithms. However, unlike those, that uses as a metaphor the survival of the better-adapted ones, in PSO the motivation is the simulation of social behavior. Such as the

other evolutionary algorithms, PSO is initialized with a population of random solutions. The main difference is that in PSO, each potential (individual) solution is also associated to a random velocity in the initialization process; the potential solutions, designated as particles, may therefore "fly" through the parameter space of the problem.

3. The Hunting Game

DAI is a field that has been quickly unfolded and diversified since its beginning in the second half of the 1970's decade. It still represents a promising research and application domain as well, characterized by its multi-disciplinary nature, gathering concepts from fields such as Computer Science, Sociology, Economy, Administration, Work Management and Philosophy.
DAI's primary focus is on the coordination as interaction form, particularly significant to reach the goal or to fulfill the tasks. And the two basic and contrasting standards of coordination are competition and cooperation. In the competition, several agents work against each others because their goals are conflicting. In the cooperation, the agents work together and they congregate their knowledge and capacities to reach a joint objective (Weiss, 2000). In this work, by the own nature of the application chosen (group hunting), only cooperative strategies will be considered.
The hunting's problem (or persecution's game) is a classical theme in AI. Just as originally proposed (Benda & Jagannathan, 1985), it is made of two classes of agents (predators and preys, classically four predators and a single prey) disposed in a rectilinear grid (plane and discrete domain), all at the same velocity. The predators' purpose is to catch the prey. To encircle it, each predator should occupy an adjacent "square" to the prey in the rectilinear grid. The prey, in turn, "wins" the game if it gets to escape away from the "board's" domain borders before being caught. In this classic version, the agents' movement is quite simple: at each "step" or simulation cycle, each agent can move a "square" in vertical or horizontal direction, since this square is not occupied yet. In general the prey is programmed with random movement, while the predators' strategy is the focus on the approaches of AI.
This kind of problem acts as an excellent benchmark for comparative evaluation on different approaches of artificial intelligence, using central, local or distributed control (Manela & Campbell, 1995). Given the nature of the problem, each individual influences and is influenced by the whole system and, since that the goal cannot be reached by a single agent separately, it is only natural the emergence of cooperative strategies. This work proposes an evolution in the game in its classical form, so that the hunt and the persecution extrapolate the discrete plane, expanding to a continuous and three-dimensional domain. Another evolution point consists of a more elaborated formulation for the behaviors and the strategies, for both predators and prey.

3.1 Behavior of Prey and Predators

In this section, the concepts for the three dimensional hunting are defined. Catching is the situation in which at least one of the predators reaches the prey. Technically speaking, it invades a kind of "vital bubble", a small sphere related to the prey's body volume and having it as the center. The prey, in its turn, has as a goal to avoid being caught by the predators for a determined period. Time limiting is due to a series of practical circumstances, among them all the fact that the performance in the game will serve, in the

end, as a target function of the algorithm's optimization; and it should naturally have a foreseen stop criterion to the cases of which predators are unable to succeed in catching the prey. The prey's behavior in this 3D pursuing game in the continuous space also evolved in relation to its classical version. It is under the control of a Finite State Machine (FSM). The projected machine to describe the prey's behavior has three states:

- Free walk: Initial state of the game. In this state, the prey behaves as it does in the classical version, having random movement at a low velocity given by $\mathbf{v}_{yw}$.
- Escape: The prey, when noticing that is being hunted, because at least a predator is inside of its "perception bubble" with velocity above certain limit given by $\mathbf{v}_{dc}$ or had it settled whatever the velocity, in its "proximity bubble", begins to move in its maximum velocity given by $\mathbf{v}_{ye}$, to a calculated direction considering the two closest predators' standings. The perception and proximity bubbles are spheres centered in the prey with radius of R_{ye} and R_{yx}, respectively (Figure 1). If the conditions that determined to get into this state have dropped away, the prey turns in to the Free Walk state.
- Caught: In this state, at least one predator has already invaded the prey's "vital bubble". It is considered being caught and its action is ceased. The game is closed. The vital bubble is a sphere centered at the prey with radius of R_{yv} (Figure 1).

Predators' behavior is controlled by a FSM set by the following three states described below:

- Hunt: Initial state of the game. The predator moves with a velocity that has direction and module defined by its strategy. This strategy's synthesis what will be exactly the target of this work, the product of the synthesis' tool that involves a simulator and an optimization algorithm.
- Pursue: The predator sets itself in this state when the prey sets in escaping mode, perceiving the approach of this one or even another predator. When pursuing, the predator has its moving direction exclusively based on the prey's position, and it moves itself at maximum velocity given by $\mathbf{v}_{dp}$, thereon, having no more any kind of strategy.
- Capture: The predator invaded the prey's "vital bubble" and its catching proceeded. The game is over.

3.2 Hunting Game's Parameters

Given the general rules, the game of three-dimensional hunt still has certain versatility margin. Several combinations can be checked and compared, depending on how some free parameters are defined; this joined to some simulation parameters arranges the configuration's settings of the game: number of predators n_d, game time limit t_{max}, initial distance D_0 and coefficient of inertia C.

The predators have to capture the prey under the considered time limit; in case this time has passed, the prey is considered as the winner. The initial distance defines the distance between the prey and each one of the predators in the beginning of the game. The coefficient of inertia must be between 0 and 1, and it simulates the inertia effect, both for the predators and prey, at each simulation step. It is applied accordingly to the following expression:

$$v(t) \leftarrow C\ v(t\text{-}1) + (1 - C)\ v(t) \tag{3}$$

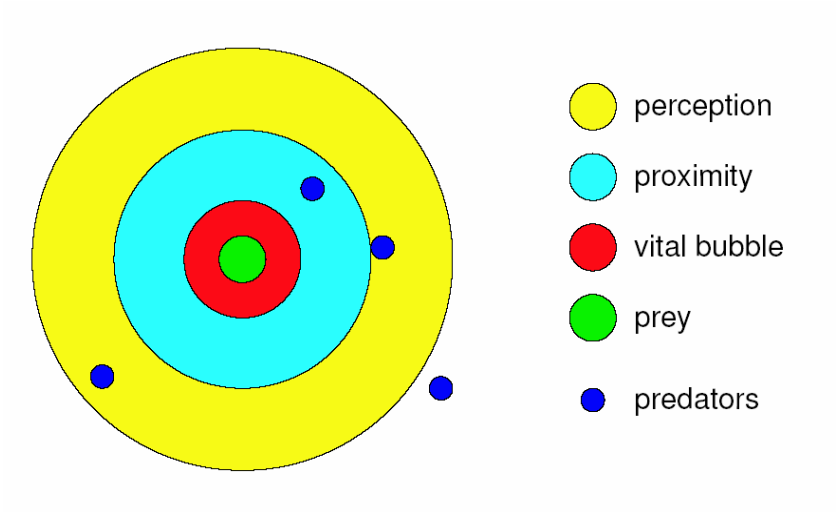


Figure 1. Perception, proximity and vital bubbles of the prey

It is natural to expect that, for each (coherent) group of free parameters, the strategies' synthesis tool presents a different result, since the circumstances aroused by these parameters in the hunt conditions be sufficiently distinct.

3.3 The Predators' Strategies

The predators do not need to move in-group, but to increase their survival chances they should be capable of self-organize in order to encircle and to capture a prey, thus guaranteeing their food. Evidently, without using any type of direct communication during the task. Police officers' teams use the radio to coordinate their encircling maneuvers, but wolves do not use walkie-talkies at the time they leave in pack for hunting a moose (Parunak, 1997). Through some rules based on visual tracks about the moose's grounding and of the other wolves, they must be able to encircle the moose without any explicit form of communication or negotiation (nothing such like "I'll go to the northward; why don't you go southward?").

All of the predators should share (as part of the problem's definition) the same hunt strategy, which does not involve memory or direct communication among the agents. Taking as a base the predators' sensorial world at the hunting time, the strategy ought to, at most, take into consideration the following parameters: direction of the prey $\mathbf{q}_y$, direction of the closest predator (neighbor) $\mathbf{q}_n$ and distance of the prey D_y. As the kinematical reference for the predators are their own positions, the first two parameters should be calculated in the following way:

$$\mathbf{q}_y = (\mathbf{P}_y - \mathbf{P}_d)/(|\mathbf{P}_y - \mathbf{P}_d|) \tag{4}$$

$$\mathbf{q}_n = (\mathbf{P}_n - \mathbf{P}_d)/(|\mathbf{P}_n - \mathbf{P}_d|)$$

Where $\mathbf{P}_y$, $\mathbf{P}_d$ and $\mathbf{P}_n$ are the vectors of position (in relation to the center of the coordinate system) of the prey, the current predator and its closest neighbor, respectively. The only

capacity of allowed decision to the predators in the game is to control its own moving, that is, dynamically change its vectorial velocity using the parameters provided by its sensorial system ($\mathbf{q}_y$, $\mathbf{q}_n$ and D_y). The strategy formulation developed for this work obeys to the following expression:

$$D = \min (D_y / D_0 , 1) \tag{5}$$

$$\mathbf{v}_d(d) = \mathbf{v}_{dp}\, d^{X1} (d^{X2}\, \mathbf{q}_y + X3\, d\, \mathbf{q}_n) / (| d^{X2}\, \mathbf{q}_y + X3\, d\, \mathbf{q}_n |)$$

Where $\mathbf{v}_d(d)$ is the velocity of the predator as a function of normalized distance to the prey; and X1, X2 and X3 are decision parameters (provided by the strategies' synthesis tool). The variable X1 is related to the influence of the predator's distance to the prey in the module of its velocity; X2 reflects the importance of the prey's direction in the vectorial composition of the velocity's direction, whilst X3 represents the influence of the closest neighbor's direction. The strategies are then defined by triple ordinates (X1, X2,X3) that define the behavior of the predators' movement. The presented formulation allows no-linearity and a wide range of possibilities both for the module and the vectorial composition of the velocity as a function of the predators' distance to the prey. Evidently, the same problem would hold other countless strategy formulations. Amplifying the sensorial capacity, for instance, the predators' velocity could also be a function of the distance to the closest neighbor. Or even maintaining the defined sensorial group in this work, quite diverse mathematical expressions could be formulated.

4. Synthesis of Strategies by the Particle Swarm

Simulation environments are important to evaluate the performance of strategies that could not be tested in any type of analytical expression. The same strategy is used by a predators' group, each influencing towards movement of another, in a chain reaction. The hunt can only be verified through the strategies' effects overtime. Based on this model, a simulation environment was implemented, and beside an optimization algorithm, it composes a synthesis tool that will provide, at the end of the process, a satisfactory solution for the initial problem. That tool was implemented in the program called PREDATOR. During the process of strategies' synthesis, the simulator changes information continuously with the optimizer. The optimizer, tracking its algorithm (particle swarm), defines some "points" in the space of solutions and it submits, one by one, these solution-candidates to the simulator, that receives and interprets them as the entry parameters for the simulation. At the end of the simulation, the outcomes of this are sent back to the optimizer, which interprets this information as the objective function of the solution-candidate that had been sent. It processes this information inside the algorithm routine and sends the next solution-candidate, restarting a cycle that only ends when the optimizer finds some of its stopping criteria. Figure 2 shows the optimization flow diagram.

To illustrate the simulation of the hunt game in the program PREDATOR, all agents were represented by fishes. These animals were chosen because, unlike humans, they are subject to huntings that can only be modeled in three-dimensional domains.

In the simulation, predators and prey's velocities possess a term of "momentum", which emulates a kind of "inertia", even without processing the dynamic equations that would necessarily involve forces and the agents' masses. This mechanism avoids that the agents (in

this case the fishes) might be moved in an implausible way in the simulation setting, executing curves and maneuvers visibly out of the dynamic reality that should reign in a real situation.

The program PREDATOR was designed in a way to present a quite simple and intuitive interface (Figure 3). All of the buttons and displays were clustered in a control panel, in which their buttons allows the user to load a file containing one of the three types of possible configurations: simulation, optimization and animation (of the particles' movement resulting from an optimization process). In the illustrated example in Figure 3, the strategy used in the simulation is the correspondent to the particle of number 1 of the 40th iteration (40-01), the line right below exhibits their parameters' strategy (X1, X2 and X3) and the inferior line presents the predators' medium distance in pursuing (the number of predators in pursuing is presented between parenthesis). There are two chronometers to the right in the panel. The first indicates the hunting [1] time; and the second, which starts only when the first one stops, marks the pursuing [2] time. There is also a field indicating the prey's current state (in the example, in escape).

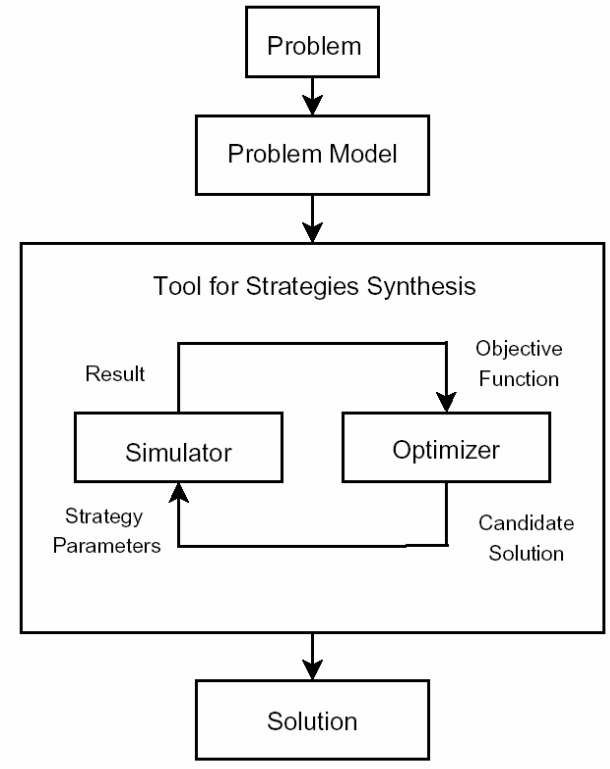


Figure 2. Optimization flow diagram

[1] All agents are in the first state of their FSMs.
[2] All agents are in the second state of their FSMs.

In spite of the interface's detailed information about positions and the fishes' velocities, to follow the simulation through the observation of tables is a superhuman task. To help the user to understand the simulation, the program PREDATOR provides four visualization windows, which exhibit the perspective projections of the tracking cameras (Figure 3). Even with all of the visualization resources implemented, it is not easy to process the visual information of the four (or even three) images in real time. Behind this difficulty there is probably a hidden evolutionary reason. The user (that is a human being) didn't have, unlike what probably happened with the fishes, his cognitive system adapted to process visual information coming from all directions. In an attempt to build a "bridge" among these differences in visual processing, a "3D radar" was implemented. This radar consists of two disks, being the superior used to represent the front part of the prey and the inferior the back part, working as if it was a rear-view mirror.

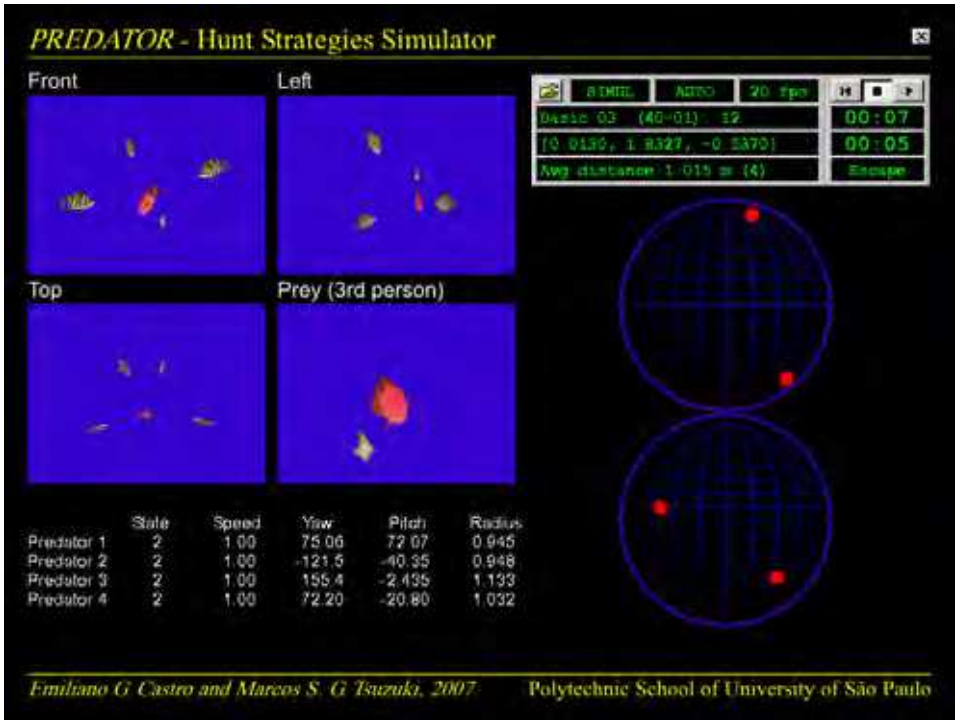


Figure 3. Window of the program PREDATOR

The optimization algorithm, "embedded" in the PREDATOR program, in order to evaluate the result of the simulation, uses the following rule that works as the objective function:

- For strategies that get to capture the prey in an available time (the free variable t_{max}), the result of the simulation is:

$$f(X1,X2,X3) = t_{hunt} + t_{purs} \quad (6)$$

Where t_{hunt} is the time used in the hunt and t_{purs} the time spent in the pursuing (both in seconds), that are exhibited in the two chronometers of the control panel;

- For strategies that don't get to capture the prey inside of the available time, the result of the simulation is:

$$f(X1,X2,X3) = t_{max} (1 + D_{average} / D_0) \quad (7)$$

Where t_{max} is the time limit allowed for the simulation (in seconds), $D_{average}$ is the medium distance of the predators to the prey at the end of the simulation and D_0 the initial distance between the predators and the prey (both in meters).

Besides the simulations' configuring information, necessary for the simulating that will evaluate the strategies generated by PSO, the program also reads the adjustment parameters in the configuration file from PSO's algorithm itself. Moreover, besides the kinematics' variables of the particles, they comprise the maximum number of iterations, the number of particles, the number of simulations by particle and the size of the neighborhood (number of particles of the subset considered as each particle's "neighborhood", including itself, for the comparison among the best individual results obtained as yet).

5. Results

Dozens of optimization tests were accomplished, varying not only the simulation's configuration features, in order to analyze different "rules" for the three-dimensional hunting game, but also the optimizing configuration, to study the sensibility of the parameters of the PSO.

The parameter "number of simulations by particle" revealed itself, soon in the preliminary tests, of fundamental importance within the optimization process. In some tests considering just a single simulation run by particle, it was common that some particles presented performance of difficult reproduction, which were resultants from simulator initializations extremely advantageous (all of the predators already actually encircling the prey, for instance). As the initialization of the predators' positions is stochastic, that is a phenomenon hard to avoid. And, as a consequence, these results got registered in the algorithm and they "pulled", in a certain measure, the swarm for a solution often far from being the most appropriate, but only "lucky" when first (and only) evaluated.

Although the PSO's algorithm is strong enough to escape from these traps, its performance gets quite harmed, once several iterations are wasted while the influence of these exceptional results does not weaken. Increasing this parameter from 1 to 5, these situations were practically eliminated, and as the performance's average of the simulations is the one to be considered as the particles' objective function, this repetition ends up working as a type of filter against the results derived from extraordinary random conditions.

The optimization's amount of time, considering as stop criterion the maximum number of iterations, is certainly a function of a series of parameters, that might be divided in two groups: the simulation and optimization one. In the first group, the most evident is the simulation time limit (45 seconds to most tests). In the second, they are preponderantly the maximum number of iterations (50 or 100), the number of particles (tests were made with 10 and 20) and the number of simulations for particle (5 in most of the tests).

The parameter t_{max} limits the time for the simulation, and consequently affects the hunting. Its value is intrinsically related to the velocity of the fishes and the several bubbles radius.

The parameter t_{max} has a fundamental influence in the objective function. A t_{max} with a small value will not allow the development of some strategies which are not so interesting initially. However, these strategies can generate very interesting solutions at the end. A larger value of t_{max} can reduce the problem; however, values too large of t_{max} will increase exaggeratedly the computational time. Finally, it is suggested that before the optimization, a sequence of preliminary simulations must be done, so that it is possible to estimate the impact of the parameters in the hunting.
A basic simulation setting was made (see Table 1), for reference, and tested with a series of preliminary tests. Variations of this basic setting were elaborated with the purpose of detecting the sensibility of some parameters.
The simulations could be watched in the Predator's visualization windows, real time. The program also renders three views of the paths described by the agents during the pursue, condensing all the animation in a single picture. An example can be seen in Fig. 4, which shows the trajectories of all agents in a successful hunt. The sudden trajectory change made by all fishes (easily seen in the path of agent number 5 in the top view) is related to the moment the fishes stop moving to encircle the prey and start pursuing it.

n_p	= 6	(number of predators)
t_{max}	= 45 s	(time limit of the game)
v_{pw}	= 0,2 m/s	(prey's walk velocity)
v_{ye}	= 1,0 m/s	(prey's escape velocity)
D_0	= 10 m	(initial distance)
R_{pe}	= 5 m	(radius of the prey's bubble of perception)
R_{pr}	= 3 m	(radius of the prey's bubble of proximity)
R_v	= 0,3 m	(radius of the prey's vital bubble)
v_{dw}	= 1,0 m/s	(predators' pursuing velocity)
v_{dp}	= 0,6 m/s	(predators' critical approach velocity)
C	= 0,95	(inertia coefficient)

Table 1. Basic scenario configuration

Some optimizer configurations were tested to generate the best strategy for this basic simulation setting, taking as base values suggested by the literature of this area (Kennedy & Eberhart, 2001). Different PSO's adjustments took to strategy parameters with very much alike performances, all providing very fast hunts. These configurations (and results) are shown in Table 2. Figure 5 allows an analysis of the convergence from four different adjustment settings of used PSO.

Configuration A has a slower convergence as a consequence of the reduced number of simulations, while configurations B and C (with very similar adjusts) presented convergence curves very similar. Configuration D showed a faster convergence, which can be a consequence of the greater number of particles allowing an efficient exploration of the parameter space.

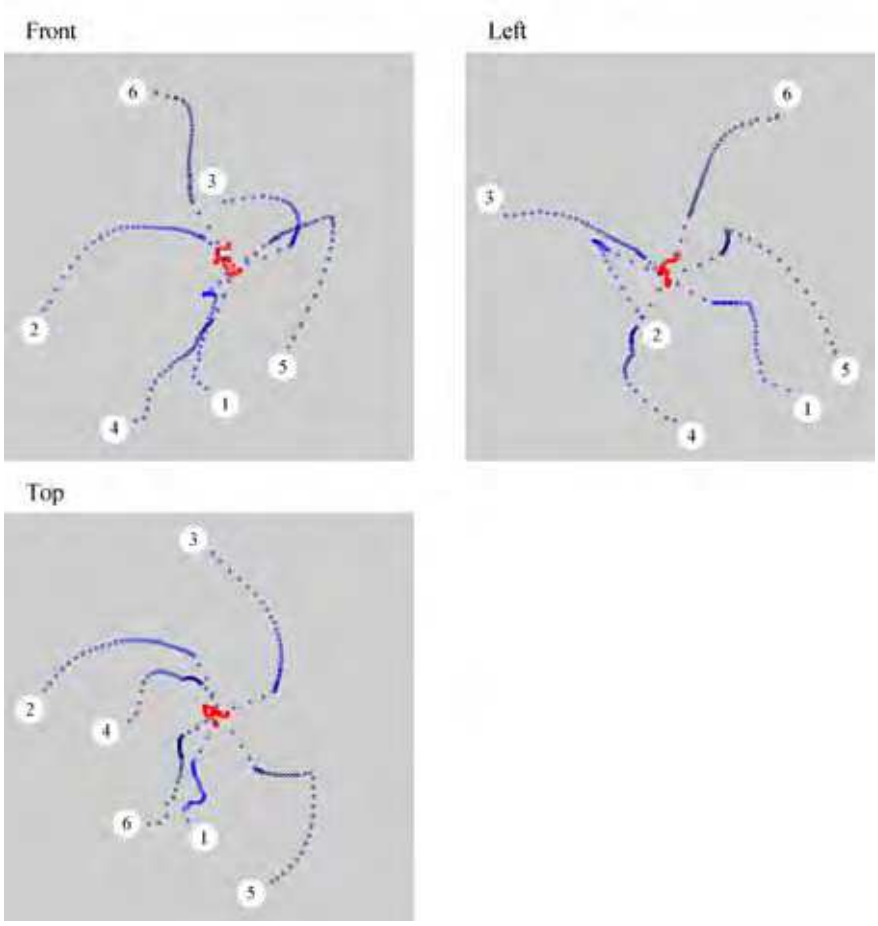


Figure 4. Trace of 6 predators (blue) hunting the prey (red)

The strategy parameters created in these tests are triple ordinates very similar, indicating "points" in the universe of solutions very close to one another. One of those tests, for instance, generated as the best strategy (for the particle number 14, in the 50th iteration) that one capable of capturing the prey after a hunt of, on average, 10.2 seconds, and defined by the parameters (X1, X2, X3) = (0.0753; 1.6763; -0.2653).

	CfgOtim A	CfgOtim B	CfgOtim C	CfgOtim D
Number of Particles	10	10	10	20
Number of Simulations/Particle	1	5	5	5
Neighborhood Size	5	5	5	5
Inertia weight (w)	1,0	1,0	1,0	1,0
φ_{1max}	0,5	0,5	0,4	0,7
φ_{2max}	0,2	0,2	0,2	0,4
V_{max}	1,0	2,0	2,0	2,0
Hunt time of the best generated strategy (s)	11,6[3]	10,4	11,4	10,2

Table 2. Generated strategies for the basic scenario

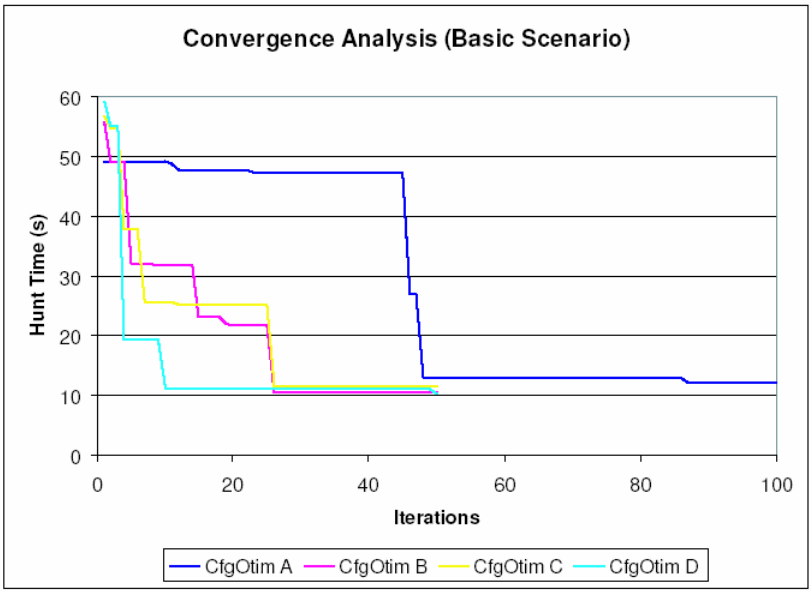


Figure 5. Convergence Analysis

Besides the results shown in Fig. 4, it is possible to visually observe the PSO optimization process through the animation mode of the PREDATOR program. Fig. 6 presents 4 frozen

[3] Value obtained at iteration number 163.

moments of an animation basic scenario, with PSO configured with 20 particles. Each frame contains three cameras illustrating the three views (top, front and rear) of the parameter space, represented as a gray box with dimensions defined by the constraints.

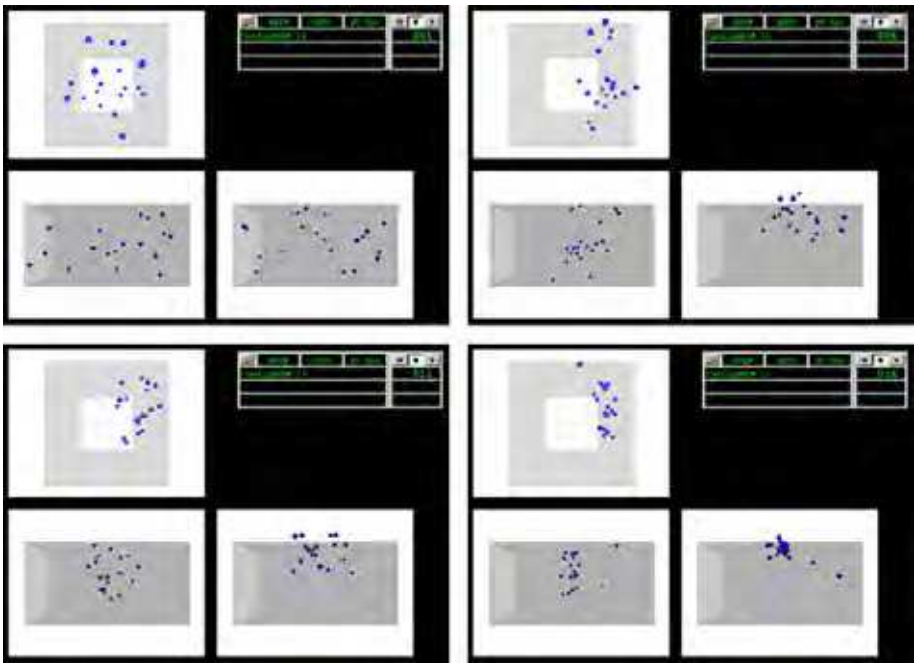


Figure 6. Twenty particles (PSO) searching for better strategies at four frozen moments: t=1 (upper left), t=6 (upper right), t=11 (lower left) and t=16 (lower right)

6. Conclusions

The problem solving capability of the multiagent systems using co-operative strategies expands the solution universe of some classes of problems to beyond some human limits of reasoning and intuition. The hard time we have trying to mentally process, in a parallel fashion, the several "instances" sensorial-cognitive-motor that represents the agents is the reason that makes invariably surprising the system's behavior as a whole. That is the emergent behavior.

In this work, we tried to demonstrate that it is possible to project these Distributed Artificial Intelligence systems knowing just how to model the problem. The "fine tuning" of the project details was accomplished by a synthesis tool, using an optimization systems that also takes advantage of the emergent behavior (of the PSO particles). Therefore, part of the problem's complexity was solved without any analytical approach, but using instead an intelligent and automatic solution searching process.

The results obtained in this work can attest that the tool of synthesis developed is really capable to provide, as long as working with well elaborated models, satisfactory solutions for problems of complex nature, of difficult resolution by analytical approaches.

Delegating the task of solving a problem for a tool in the moulds proposed in this work means, in a final analysis, to trust in the emergent properties of a complex system to produce solutions for another system (not coincidentally, also complex). These solutions are not pre-defined in the program. Neither are we capable to easily understand the solutions generated in terms of the program code. This tends to go in the opposite direction of the essence from traditional engineering. However, will eventually unveils a field still poorly explored in the area of project development.

7. References

Benda, R. D. M. & Jagannathan, V. (1985) On optimal cooperation of knowledge sources. In *Technical Report BCS-G2010-28*, Boeing AI Center, Boeing Computer Services, Bellevue. Washington.

Huhns, M. N. & Stephens, L. M. (1999) Exploiting expertise through knowledge networks. *IEEE Internet Computing*, Vol. 3, Number 6, pp. 89-90.

Kennedy, J. & Eberhart, R. C. (2001) *Swarm Intelligence*. Ed. Morgan Kaufmann, Sao Francisco.

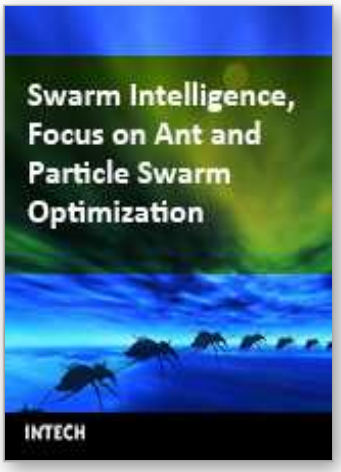
Magoulas, G. D.; Eldabi, T.; Paul, R. J. (2002) Global search for simulation optimisation. *Proceedings of the 2002 Winter Simulation Conference*, Ed. Ycesan, E.; Chen, C. H.; Snowdon, J. L.; Charnes, J. M. pp. 1978-1985.

Manela, M. & Campbell, J. A. (1995) Designing good pursuit problems as testbeds for distributed AI: a novel application of genetic algorithms. In *Lecture Notes on Artificial Intelligence 957*, Springer Verlag.

Parunak, H. V. D. (1997) Go to the ant: Engineering principles from natural agent systems. In *Annals of Operations Research*.

Weiss, G. (2000) Multiagent *Systems: a modern approach to Distributed Artificial Intelligence*. Ed. MIT Press, Massachusetts.

Wooldridge, M. J. & Jennings, N. R. (1999) Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, Vol. 3, Number 3, pp. 20-27.



Swarm Intelligence, Focus on Ant and Particle Swarm Optimization

Edited by FelixT.S.Chan and Manoj KumarTiwari

ISBN 978-3-902613-09-7
Hard cover, 532 pages
Publisher I-Tech Education and Publishing
Published online 01, December, 2007
Published in print edition December, 2007

In the era globalisation the emerging technologies are governing engineering industries to a multifaceted state. The escalating complexity has demanded researchers to find the possible ways of easing the solution of the problems. This has motivated the researchers to grasp ideas from the nature and implant it in the engineering sciences. This way of thinking led to emergence of many biologically inspired algorithms that have proven to be efficient in handling the computationally complex problems with competence such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), etc. Motivated by the capability of the biologically inspired algorithms the present book on "Swarm Intelligence: Focus on Ant and Particle Swarm Optimization" aims to present recent developments and applications concerning optimization with swarm intelligence techniques. The papers selected for this book comprise a cross-section of topics that reflect a variety of perspectives and disciplinary backgrounds. In addition to the introduction of new concepts of swarm intelligence, this book also presented some selected representative case studies covering power plant maintenance scheduling; geotechnical engineering; design and machining tolerances; layout problems; manufacturing process plan; job-shop scheduling; structural design; environmental dispatching problems; wireless communication; water distribution systems; multi-plant supply chain; fault diagnosis of airplane engines; and process scheduling. I believe these 27 chapters presented in this book adequately reflect these topics.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Emiliano G. Castro and Marcos S. G. Tsuzuki (2007). Simulation Optimization Using Swarm Intelligence as Tool for Cooperation Strategy Design in 3D Predator-Prey Game, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, FelixT.S.Chan and Manoj KumarTiwari (Ed.), ISBN: 978-3-902613-09-7, InTech, Available from: http://www.intechopen.com/books/swarm_intelligence_focus_on_ant_and_particle_swarm_optimization/simulation_optimization_using_swarm_intelligence_as_tool_for_cooperation_strategy_design_in_3d_preda

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821