A Novel Space Filling Curves Based Approach to PSO Algorithms for Autonomous Agents

Doina Logofătu[1] ✉, Gil Sobol[2], Daniel Stamate[3], Kristiyan Balabanov[1]

[1] Computer Science Department of Frankfurt University of Applied Sciences, Nibelungenplatz 1, 60318, Frankfurt am Main, Germany
[2] Industrial Engineering, Technion - Israel Institute of Technology
[3] Department of Computing, Goldsmiths College, University of London, London SE146NW, UK
logofatu@fb2.fra-uas.de

Abstract. In this work the swarm behavior principles of Craig W. Reynolds are combined with deterministic traits. This is done by using leaders with motions based on space filling curves like Peano and Hilbert. Our goal is to evaluate how the swarm of agents works with this approach, supposing the entire swarm will better explore the entire space. Therefore, we examine different combinations of Peano and Hilbert with the already known swarm algorithms and test them in a practical challenge for the harvesting of manganese nodules on the sea ground with the use of autonomous robots. We run experiments with various settings, then evaluate and describe the results. In the last section some further development ideas and thoughts for the expansion of this study are considered.

Keywords: autonomous agents, space filling curves, particle swarm optimization, deterministic leaders, application

1 Introduction

Simultaneously with the applied research on renewable resources, it is useful to find novel ways for opening up fossil ones. As example, manganese nodules can be found on the sea bottom. A considerable application field involves rust and corrosion prevention on steel [9,10]. The degradation could be reduced substantially by collecting these manganese nodules from the sea bottom using specialized autonomous agents. Our focus in this work is to evaluate different ways in handling the movement of these agents. The experiments can be extended to cover other collecting tasks. The base for our application is a framework for simulation and improvement of swarm behavior in changing environments [1], which we redesign and extend. It simulates the swarm behavior by using the principles of Craig W. Reynolds [2]. The main purpose of the framework regarding the application is to deploy agents with a specific strategy and then to gather them. While gathering, the agents are collecting the manganese which is distributed on every position in the coordinate system. Once gathered together, there is no more movement and the simulation ends. Naturally manganese occurs in form of nodules, thus it is distributed uniformly. For the different forms of the manganese distributions, we created several benchmarks used in the results' comparison. The next step of improvement would be the collecting procedure. The greater distance the agents move, the higher is the probability to find manganese. Consequently, we intend to reach a way for passing through a larger area. The easiest solution would be to define for each agent its own path. This would probably scatter the swarm because of the bad orientation, the changing environment and the uneven surface. Most of the research works regarding swarm behavior are inspired by nature like genetic algorithms or particle swarm optimization. These outcomes focus on fish schools or bird flocks. An alternative discussion could consider, for example, a pack of wolves. A pack of wolves means actually autonomous individuals with a specific hierarchy. Not every wolf has the same power regarding decisions for the pack. Normally there is one wolf who leads the group and the others are followers [11]. This contribution aims to study this notion more closely. We intend to set one or more leaders who will move after a given route, but still be part of the swarm, and the rest calculate their new position, that means every iteration in consideration of all agents.

2 Background

This section introduces previous work the application is based on, followed by three main topics: Moving Algorithms, Particle Swarm Optimization, Hilbert and Peano Curves.

2.1 Framework for Adaptive Swarms Simulation and Optimization

The application we consider first is based on [1]. The framework is an application that runs a simulation of autonomous robots using moving algorithms Random, Square, Circle, Gauss, and Bad Centers [1]. It contains several fundamental deployment strategies used from where the moving algorithms start. The front end uses the open source framework of *processing.org* [4]. It creates the chosen deployment strategies and calculates the movement of the autonomous agents, as well as the collection of manganese nodules. In addition, the number of agents can be settled and it counts the distance in walked meters of all agents together.

2.2 Moving Algorithms

In our practical application, it is required to build a swarm of autonomous agents, where each agent individually moves forward taking into consideration the other agents of the group. There are efficient algorithms for swarm behavior and movement of agents that are implemented in the application [6]. The previous work [1] uses a simplification of the bird flock movement described by Craig W. Reynolds [2]. The contribution implemented three different algorithms that run simultaneously: cohesion, separation, alignment.

2.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was introduced in 1995 by J. Kennedy and R. Eberhart [6]. The innovation was building swarm behavioral approaches for solving problems by iteratively improving a candidate's solution until termination criteria is satisfied [7]. It is similar to a genetic algorithm as both algorithms are initialized with a random population, in PSO called particles. The difference is that in PSO algorithms, each particle is assigned to a randomized velocity and the particles move through hyperspace. Each particle is defined by its position, velocity, current objective value and personal best value of all time. PSO also keeps track of the global best value that is the best objective value of all particles and also the corresponding position.

2.4 Space Filling Curves

A Space Filling Curve is a special function of calculus that fully covers a two or three dimensional space. Giuseppe Peano (1858-1932) discovered them first in 1890. He wanted to create a continuous mapping construction from the unit interval onto the unit square [7].

Peano Curve. Till 1890 one assumed that a constant curve with parametric function of only one variable $x = \phi(t)$ and $y = \psi(t)$, cannot reflect surjectively the unit interval onto the unit square. The reason for this was the theorem of Eugen Natto, who showed that a bijection must be unsteady to satisfy this. However, Peano found a steady function f_p, such that $f_p(I) = 2$.

Definition 1. *Peano Curve [10]. The projection $f_p : I \rightarrow 2$ with*

$$f_p(0_3, t_1t_2t_3t_4) = \begin{pmatrix} 0_3 \; t_1(k^{t_2}t_3)(k^{t_2+t_4}t_5) \; ... \\ 0_3 \; (k^{t_1}t_2)(k^{t_1+t_3}t_4) \quad ... \end{pmatrix} .$$

and the operator $k^{t_j} = 2t_j (t_j = 0, 1, 2)$, where k^v is the v-th iteration of k, we call Peano Curve.

So according to this definition we have:

$$f_p(0_3, 00t_3t_4...) = \begin{pmatrix} 0_3, \; 0\xi_2\xi_3\xi_4 \; ... \\ 0_3, \; 0\eta_2\eta_3\eta_4 \; ... \end{pmatrix} .$$

To create the Peano's curve we start at point $(0, 0)$ and finish in the diagonal corner at point $(1, 1)$. The starting point of a sub square must be the endpoint of the previous sub square. Figure 1 illustrates where the start- and endpoints are with the use of arrows.

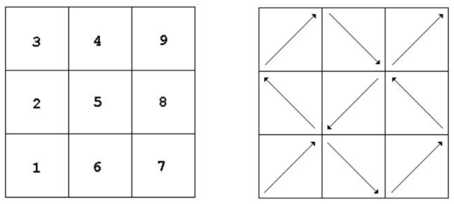

Fig. 1. Construction order and orientation for the Peano Curve.

Hilbert Curve. Peano (1890) introduced first the space-filling curves. Hilbert (1891) popularized their existence and gave an insight into their generation. His statement was that if the unit interval can be mapped steady onto the unit square, then also sub intervals can be mapped steadily onto sub squares. In the first step, Hilbert divided the unit interval into four sub intervals of the same size as well as the unit square into four equally sized sub squares, where each sub interval is mapped onto one sub square [7].

3 Implementation Details

This section describes shortly the practical changes and extensions that were necessary to implement for the experimental procedure. At first, some new classes had to be implemented to lay the basis for the new *Manganese-Nodule-Model*. These new classes help us to represent the nodules on the map and for background calculations as well as. Some of the new implemented classes and interfaces include: *DeployRing*, *DeploymentStrategy*, *ManganeseNodule*, *VisualManganeseNodule*. The used benchmarks for the manganese nodules distribution are independent files. Each line represents a y-value and each char represents an x-value in the coordinate system of the graphical user interface. The lines are filled with numbers from 0 to 7 in accordance with the size of the nodule, where zero means that no nodule can be found on this position.

3.1 Peano/Hilbert Algorithms

The Peano algorithm is implemented using a recursive function, that is called every time when the agent moves into the next unit square. The function calls change from clockwise rotation to negative rotation (counterclockwise). The implementation of the Hilbert Algorithm is analogous to the Peano Algorithm (Alg. 1). The basic structure of how the exploring through the sub squares is done is fixed.

4 Experimental Results

This section presents relevant results we achieved with the extended implementations to the application. The distance and the collected amount of manganese

Algorithm 1: Pseudo Code Peano Algorithm

```
peanoAlgorithm(length, direction, rotation, deep){
if under lowest level then
|   return
end
        peanoAlgorithm(length, direction, clockwise rotation, deep-1)
        step forward with given length and direction

        peanoAlgorithm(length, direction, counterclockwise rotation, deep-1)
        step forward with given length and direction

        peanoAlgorithm(length, direction, rotation, deep-1)
        direction turn clockwise with given rotation degree
        step forward with given length and direction

        direction turn clockwise with given rotation degree
        peanoAlgorithm(length, direction, counterclockwise rotation, deep-1)
        step forward with given length and direction

        peanoAlgorithm(length, direction, rotation, deep-1)
        step forward with given length and direction

        peanoAlgorithm(length, direction, counterclockwise rotation, deep-1)
        direction turn counterclockwise with given rotation degree
        step forward with given length and direction

        direction turn counterclockwise with given rotation degree
        peanoAlgorithm(length, direction, rotation, deep-1)
        step forward with given length and direction

        peanoAlgorithm(length, direction, counterclockwise rotation, deep-1)
        step forward with given length and direction

        peanoAlgorithm(length, direction, rotation, deep-1)
}
```

of all agents in one pass are the examined variables. The difference of agents between *Rob Total* and the sum of *Rob Hilbert* and *Rob Peano* are robots behaving according to the principles of typical Moving Algorithms.

4.1 Diamond, Square, Peano 0-50

In this experiment we increased the number of Peano Robots and ran 1000 iterations with every increase. This experiment runs with the benchmark Diamonds and the deployment strategy Square. With every increase of the number of Peano Robots, the covered distance of all robots increases by 30 000 m to 50 000 m with an average increase of 46 081.46 m. The collected manganese does not increase constantly. The global maximum of 53 080 kg is reached with a constellation of 44 Peano Robots (see Fig.2, left). The biggest discrepancy of 15 is between the first and the second measurement, with an increase of 589 %. The fewer meters a robot has to travel for the same amount of manganese, the more efficient it is. The second diagram in Fig.2 (right) shows this relation of average distance per kg manganese for each number of Peano Robots. The best efficiency occurs without any Peano robot in the simulation with an average distance of $3\,\mathrm{m\,kg^{-1}}$ manganese. But as we can see in the other diagram (Fig.2, left) the total amount of manganese is very little. So we want to focus on analyzing all cases where Peano robots are involved. There are a few amounts of Peano robots with a very close distance per kg manganese. This is the case with the amount of 2, 3, 4, 5 and 6 Peano robots (average absolute deviation 3.9 m), with the amount of 16 to 21 Peano robots (average absolute deviation 3 m) or with the amount of 36 to 42 Peano robots (average absolute deviation 2.1 m). Another interesting point can be seen at 44 Peano Robots where the total manganese maximum is. The distance per kg manganese diagram shows here a local minimum of $383\,\mathrm{m\,kg^{-1}}$ manganese. This leads to the conclusion that we have a reasonably efficient constellation.

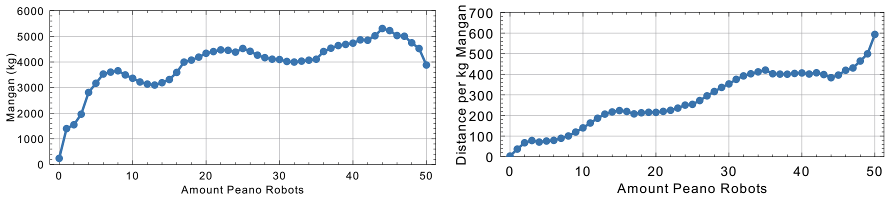

Fig. 2. Analysis of Diamonds, Square, Peano 0-50 increase: collected amount of manganese for (left); relation between the total amount of collected manganese and the distance all robots have covered.

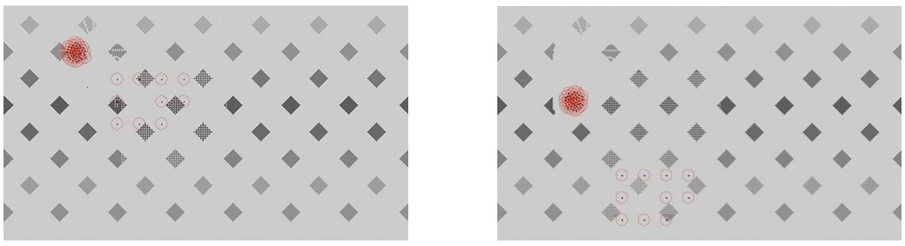

Fig. 3. Diamond, Square, Peano 0-50: Screenshots experimental procedure after 500 iterations (left) resp. after 1000 iterations (right).

To run the simulation only with Hilbert Robots brings unsatisfactory results. In this case, the total amount of manganese breaks down roughly 21 % compared to the simulation run with 49 Hilbert Robots.

4.2 Lines, Square, Peano 0-50

In this experiment we switched our benchmark to the benchmark Lines. The deployment strategy is Square and we increase the number of Peano Robots from 0-50.

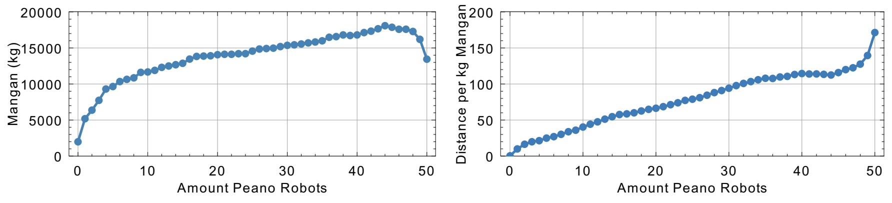

Fig. 4. Analysis of Lines, Square, Peano 0-50 increase: collected amount of manganese (left); relation between the total amount of collected manganese and the distance all robots have covered (right).

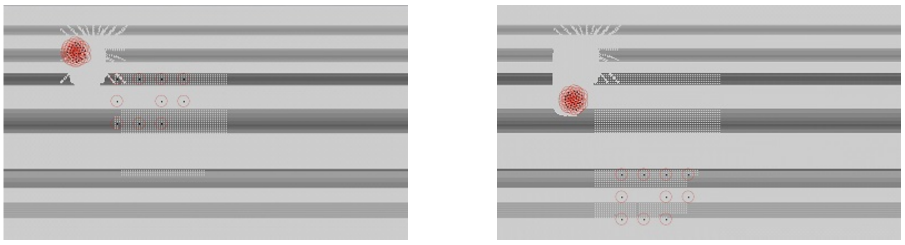

Fig. 5. Lines, Square, Peano 0-50: Screenshots experimental procedure after 500 iterations (left) resp. after 1000 iterations (right).

With every increase of the number of Peano Robots, the covered distance of all robots increases by 30 000 m to 50 000 m with an average increase of 46 081.46 m. This is identical to the results in Table 1. The collected manganese increases constantly up to an amount of 38 Peano Robots. The global maximum of 18 090 kg is reached with a constellation of 44 Peano Robots (see Fig. 4, left). With an amount of 6 Peano Robots we achieve a result of 10 341 kg total manganese. This is more than 50 % from what we achieve with our global maximum with 44 Peano Robots. That means, we have achieved half of the global maximum with an efficiency of 26.99 $\mathrm{m\,kg^{-1}}$ manganese in contrast to 112.48 $\mathrm{m\,kg^{-1}}$ manganese. In conclusion we get 100 % more manganese for 416.75 % less efficiency. That is in no reasonable relation to the benefits. Overall the distance per kg manganese increases almost linearly up to the global maximum of total manganese with 44 Peano Robots and goes steeply up afterwards. It is striking that this experiment has its maximum with the same amount of Peano Robots. The only thing that distinguishes these two experiments are the used benchmark maps.

4.3 Diamond, Square, Peano 1-25-1, Hilbert 1-49

In this experiment we mixed Hilbert and Peano Robots as well as robots using the moving Algorithms described by [2]: Cohesion, Separation and Alignment. We begin with increasing the amount of both Peano and Hilbert Robots from 0 up to 25. Then all robots are either Peano or Hilbert Robots. From this point we decrease the amount of Peano Robots and keep increasing the amount of Hilbert Robots. This experiment runs with the benchmark Diamonds and the deployment strategy Square. A total of 1000 iterations were ran with every increase. The results are presented in Table 1.

On the whole the results of this experiment are related to the two experiments we did before. The Total Mangan diagramm (Fig. 6, left) shows a rapid growth in the first 25 iterations. This is justified because every time we increase the amount of robots by two, one Hilbert Robot and one Peano Robot. For the next 15 iterations (25-10 Peano & 25-40 Hilbert) the total collected amount of mangan is more or less even. This leads to the conclusion that both algorithms have a similar efficiency, as apparently the proportion of the robots between Peano and Hilbert are distributed, as long as there is a minimum of 20% of the other programmed Robots. If we pass this 20%, the efficiency breaks down very fast and there is a result of 100 more meter per kg mangan for the proportion of 1 Peano/49 Hilbert to 10 Peano/40 Hilbert. The latter marks also the maximum of the overall amount of collected mangen with 6078 kg. These are approximately 700 kg more mangan than running only one of the algorithms.

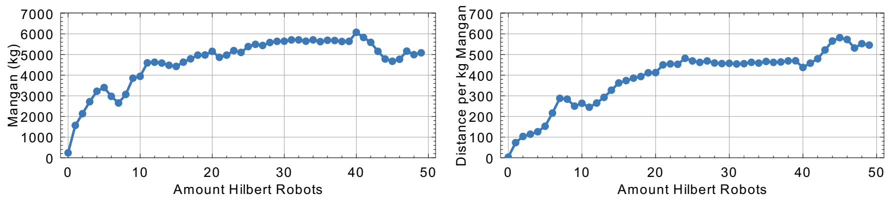

Fig. 6. Analysis of Diamonds, Square, Peano 1-25-1, Hilbert 1-49 increase: collected amount of manganese (left); relation between the total amount of collected manganese and the distance all robots have covered.

5 Conclusions and Future Work

This work focuses only on the idea to combine swarm behavior with deterministic leaders by using space filling curves. There are still many more things to try in

Table 1. Diamond, Square, Peano, Hilbert

Exp.	**Benchmark**	**Deploy. St.**	**Robots**	**Peano**	**Hilbert**	**Mangan**	**Distance**
102	Diamond	Square	50	0	0	237 kg	673.03 m
103	Diamond	Square	50	1	1	1567 kg	114,910.56 m
104	Diamond	Square	50	2	2	2133 kg	220,147.01 m
105	Diamond	Square	50	3	3	2715 kg	310,008.24 m
106	Diamond	Square	50	4	4	3222 kg	406,158.29 m
107	Diamond	Square	50	5	5	3402 kg	519,118.26 m
108	Diamond	Square	50	6	6	2976 kg	644,629.99 m
109	Diamond	Square	50	7	7	2652 kg	763,532.13 m
110	Diamond	Square	50	8	8	3068 kg	871,160.00 m
111	Diamond	Square	50	9	9	3854 kg	963,823.71 m
112	Diamond	Square	50	10	10	3944 kg	1,040,938.41 m
113	Diamond	Square	50	11	11	4597 kg	1,125,638.16 m
114	Diamond	Square	50	12	12	4628 kg	1,226,243.96 m
115	Diamond	Square	50	13	13	4587 kg	1,341,671.41 m
116	Diamond	Square	50	14	14	4480 kg	1,466,358.73 m
117	Diamond	Square	50	15	15	4424 kg	1,602,618.82 m
118	Diamond	Square	50	16	16	4631 kg	1,730,787.65 m
119	Diamond	Square	50	17	17	4792 kg	1,847,645.98 m
120	Diamond	Square	50	18	18	4971 kg	1,955,750.29 m
121	Diamond	Square	50	19	19	4978 kg	2,049,028.65 m
122	Diamond	Square	50	20	20	5155 kg	2,126,192.66 m
123	Diamond	Square	50	21	21	4866 kg	2,189,012.96 m
124	Diamond	Square	50	22	22	4972 kg	2,260,848.88 m
125	Diamond	Square	50	23	23	5190 kg	2,350,150.97 m
126	Diamond	Square	50	24	24	5102 kg	2,458,013.63 m
127	Diamond	Square	50	25	25	5386 kg	2,525,939.83 m
128	Diamond	Square	50	24	26	5494 kg	2,538,747.79 m
129	Diamond	Square	50	23	27	5439 kg	2,550,909.82 m
130	Diamond	Square	50	22	28	5583 kg	2,562,398.69 m
131	Diamond	Square	50	21	29	5640 kg	2,573,282.37 m
132	Diamond	Square	50	20	30	5646 kg	2,583,499.01 m
133	Diamond	Square	50	19	31	5709 kg	2,593,153.25 m
134	Diamond	Square	50	18	32	5709 kg	2,602,156.73 m
135	Diamond	Square	50	17	33	5646 kg	2,610,406.45 m
136	Diamond	Square	50	16	34	5709 kg	2,617,783.89 m
137	Diamond	Square	50	15	35	5625 kg	2,624,155.36 m
138	Diamond	Square	50	14	36	5691 kg	2,629,374.63 m
139	Diamond	Square	50	13	37	5679 kg	2,635,330.66 m
140	Diamond	Square	50	12	38	5631 kg	2,642,137.89 m
141	Diamond	Square	50	11	39	5640 kg	2,649,930.41 m
142	Diamond	Square	50	10	40	6078 kg	2,658,829.66 m
143	Diamond	Square	50	9	41	5826 kg	2,668,914.30 m
144	Diamond	Square	50	8	42	5590 kg	2,680,376.23 m
145	Diamond	Square	50	7	43	5157 kg	2,692,600.50 m
146	Diamond	Square	50	6	44	4784 kg	2,705,362.02 m
147	Diamond	Square	50	5	45	4674 kg	2,718,559.67 m
148	Diamond	Square	50	4	46	4771 kg	2,732,079.43 m
149	Diamond	Square	50	3	47	5165 kg	2,745,848.07 m
150	Diamond	Square	50	2	48	4994 kg	2,759,815.88 m
151	Diamond	Square	50	1	49	5084 kg	2,773,938.04 m

order to go deeper into this topic, especially using the applied research. For example, we can consider leaders with varied power (weight), i. e. the one who collected the most manganese the last time (number of iterations, seconds) will get the highest weight when calculating the next position of each autonomous agent of the group. Additionally, a distributed system could be considered and thereby the communication between the agents would be intensified. In order to get as much manganese as possible, the swarm could divide and follow different leaders or we can vary the number of agents following a specific leader. If the leader loses power, some agents can join another swarm. The leader stays on his deterministic path: thus the chance to find undetected manganese fields remains high. If a leader does not find anything for a long time, he may become a follower and join a swarm. This could also be possible the other way around. If there is a big swarm, new leaders could be chosen to search in a specific direction. Another extension would be to integrate deterministic motions with genetic algorithms, for instance build populations with different amounts of Hilbert, Peano and swarm agents, like this work already did, but develop the next generation using the principles of genetic algorithms

References

1. Canyameres S., Logofătu D.: Platform for Simulation and Improvement of Swarm Behavior in Changing Environments. 10^{th} International Conference Artificial Intelligence Applications and Innovations, AIAI 14, Springer LNCS, Island of Rhodes, Greece (2014)
2. Reynolds W.: Boids (simulated flocking). `http://www.red3d.com/cwr/boids` [Accessed 15-June-2017]
3. Shyr W.-J.: Parameters Determination for Optimum Design by Evolutionary Algorithm. DOI: 10.5772/9638 [Accessed 15-June-2017]
4. Fry B., Reas C.: Processing. `https://processing.org/` [Accessed 15-June-2017]
5. Rodriguez F., Garcia-Martinez C.: An Artificial Bee Colony Algorithm for the Unrelated Parallel Machines Scheduling Problem. PPSN XII (II), 143–152, Springer, Taormina (2012)
6. Kennedy J., Eberhart R.: Particle swarm optimization. IEEE Conference on Neural Networks, 4:1942–1948.
7. Barnsley M. F.: Fractals Everywhere. Dover Books on Mathematics, New Edition, ISBN 978-0486488707 (2012)
8. Detailed requirements for the first prototype. `http://informaticup.gi.de/fileadmin/redaktion/Informatiktage/studwett/Aufgabe_Manganernte_.pdf` [Accessed 15-June-2017]
9. Rossum J. R.: Fundamentals of Metallic Corrosion in Fresh Water. `http://www.roscoemoss.com/wp-content/uploads/publications/fmcf.pdf` [Accessed 15-June-2017]
10. Min Jun Kim, Jung Gu Kim: Effect of Manganese on the Corrosion Behavior of Low Carbon Steel in 10 wt.% Sulfuric Acid. Int. J. Electrochem. Sci., 6872–6885, 10 (2015)
11. Muro C., Escobedo L., Spector L., Coppinger R. P.: Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations. Behavioral Processes, Vol. 88, Issue 3, 192–197 (2011)