Observing the Swarm Behaviour during Its Evolutionary Design

Laura Diosan, Mihai Oltean
Department of Computer Science
Faculty of Mathematics and Computer Science
Babes-Bolyai University
Kogalniceanu 1, Cluj-Napoca, 400084
Romania.
lauras, moltean@cs.ubbcluj.ro

ABSTRACT

Evolutionary Algorithms (EAs) can be used for designing Particle Swarm Optimization (PSO) algorithms that work, in some cases, considerably better than the human-designed ones. By analyzing the evolutionary process of design PSO algorithm we can identify different swarm phenomena (such as patterns or rules) that can give us deep insights about the swarm's behaviours. The observed rules can help us to design better PSO algorithms for optimization. In this paper we investigate and analyze swarm phenomena by looking to process of evolving PSO algorithms. Several interesting facts are inferred from the strategy evolution process (the particle quality could influence the update order, some particles are updated more frequently than others are, the initial swarm size is not always optimal).

Categories and Subject Descriptors

I.2.6 [**Learning**]; I.2.8 [**Problem Solving, Control Methods and Search**]

General Terms

Algorithms

Keywords

Particle Swarm Optimization, Swarm Rules, Evolutionary Computation, Function Optimization, Meta Genetic Algorithms

1. INTRODUCTION

Various evolutionary and non-evolutionary methods were proposed for solving complex search and optimization problems. Among these methods, a special place is occupied by Evolutionary Algorithms (EAs) [9, 7] and techniques based on Swarm Intelligence (such as Particle Swarm Optimization (PSO) [6] and Ant Colony Optimisation [5]). The main advantage of these methods is given by the possibility of using them for searching in various spaces without performing big changes in the algorithm's structure. They can be easily adapted (by the human or by itself) to the particularities of the problem being solved.

PSO is a population based stochastic optimization technique proposed by Kennedy and Eberhart [12, 13, 14]. Standard PSO algorithm randomly initializes a group of particles (solutions) and then searches for optima by updating all particles along a number of iterations. In any iteration, each particle is updated by following few simple rules [11, 22].

Standard model implies that particles are updated synchronously [13]. This means that the current position and speed for a particle are computed taking into account only the information from the previous iteration of particles.

The model investigated in this paper is a more general one. We focus our analysis on an asynchronous version of the PSO algorithm. This variant has the following characteristics:

- When a particle is updated the current state of the swarm (the position and the velocities of all particles) is taken into account. The best global and local values are computed for each particle which is about to be updated, because the previous modifications could affect these two values. This is different from the standard PSO algorithm (or synchronous PSO algorithm [13]) where the particles were updated taking into account only the information from the previous iteration (modifications performed so far by a standard PSO in the current iteration had no influence over the modifications performed further in the current iteration) and it is more closely to the asynchronous PSO algorithm [1, 15] that updates particle positions and velocities continuously, based on currently available information.

- In our model the particles are updated based on their quality. This fitness-based update is important because it could be better to firstly modify the best particles of the swarm and than the worst particles (or vice versa). This is again different from the standard asynchronous PSO algorithm [1, 15] which updates particle positions and velocities using always the same predefined order: first particle, second particle and so on (there are no relationships between the update order and the particle's quality).

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

- Some particles may be updated more often than other particles. For instance, in some cases, is more important to update the best particles several times per iteration than to update the worst particles.
- During the evolution, the swarm size can be modified due to at least two reasons: some particles perform more moves (for improving their quality), while other particles are never updated. This is why the weakest particles are eliminating from the swarm. Unlike the standard PSO algorithm, which works with a preestablished swarm size, the current model finds by itself the optimal size of the swarm along the evolution.

We intend to study how we can obtain better PSO algorithms. For achieving this goal we employ an evolutionary approach: we start with a population of randomly generated PSO algorithms and we try to improve them along with a fixed number of generations. During the evolution we try to discover new swarm phenomena such as the special relationships between particles, their quality and their update order, the optimal swarm size or some rules in the update strategy of the swarm during the evolution process. These rules can be repeatedly applied for obtaining better approximations of the solution. The process of particle update is influenced by these rules.

Because a PSO is a complex algorithm, we cannot evolve all its aspects. We are taking into account only the next important one: the order in which the particles are updated. Other aspects, such s the equation used for updating a particle have been analyzed in [21]. The new swarm phenomena investigated in this paper are:

- update frequencies - how many times a particle is updated,
- order of updates - the order of particles that is taken into account for modifying the position and velocity of a particle
- optimal swarm size.

These information can help us to design better PSO algorithms for optimization.

The paper is structured as follows: Section 2 provides a brief review of the work on PSO parameters optimization. Section 3 describes the model for evolving the PSO update strategy. In Section 4.1 an analysis of the optimal swarm size detected during the evolution is presented. In addition, the frequency of updates performed into the swarm is investigated. Several rules identified in the PSO update strategy are presented in Section 4.3. Section 4.4 summarizes the most important ideas of this analysis and the main features of the developed model. Conclusions and further work directions are suggested in Section 5.

2. RELATED WORK

Many improvements to the basic form of PSO have been proposed and tested in the literature [3, 10, 18]. Also, several analysis of the PSO algorithm behaviour were performed [2, 17, 19, 21]. Much of this work is focused on the convergence.

Ozcan and Mohan [19] analyzed the trajectory of a particle in the "original" PSO algorithm (without an inertia weight or a constrict coefficient) and van der Bergh carried out the first PSO convergence study [23]. Later, Clerc and Kennedy [2] proposed the model based on the constrict coefficient.

Langdon et al. [17] evolved kernel functions which describe the average behaviour of a swarm of particles as if it was responding as a single point moving on a landscape transformed by the kernel. The evolved functions (obtained with Genetic Programming technique) give another landscape, which is "perceived" by a simple hill climber. The goal for the Genetic Programming is to evolve a kernel, which causes the hill climber to move to resemble movement of the whole PSO swarm.

Several approaches [8, 16, 20, 21, 25, 26] proposed various hybrid evolutionary algorithm that combines the concepts of EA and PSO.

For instance, Poli [21] studied the possibility of evolving, through the use of Genetic Programming, the optimal force generating equations to control the particles in a PSO (forces that stimulate each particle to fly back both towards the best point sampled by it an towards the swarm's best).

3. THE MODEL FOR EVOLVING THE UPDATE STRATEGY OF PARTICLES

The main idea of the model proposed in [4] is to evolve arrays of integers, which provide a meaning for updating the individuals within a PSO algorithm during iteration. The model is a hybrid technique that works at two levels: the first (macro) level consists in a steady-state genetic algorithm (GA) whose chromosomes encode the update strategy of PSO algorithms. In order to compute the quality of a GA chromosome a PSO algorithm (whose update order is encoded into that chromosome) is run. Thus, the second (micro) level consists in a modified PSO algorithm that computes the quality for a GA chromosome.

3.1 Representation

Standard PSO algorithm works with a group of particles (solutions) and then searches for optima by updating them during iteration.

During iteration, following two "best" values, each particle is updated. The first one is the location of the best solution that a particle has achieved so far. This value is called *pBest*. Another "best" value is the location of the best solution that any neighbour of a particle has achieved so far. This best value is a neighbourhood best and called *nBest*.

In a standard PSO algorithm, all particles will be updated once during the course of iteration. In real-world swarm (such as flock of birds), not all birds update their position and velocity in the same time. Some of them update these values more often and others update its later or not at all. In this case it is interesting to discover (evolve) a model which can tell us which particles/birds must be updated and which is the optimal order for updating them.

A GA [7] is used (in [4]) for evolving the update strategy of a PSO algorithm. Each GA individual is a fixed-length string of genes. Each gene is an integer number, in the interval $[0, SwarmSize - 1]$. These values represent indexes of the particles that will be updated during PSO iterations. Some particles could be updated more often and some of them are not updated at all. Therefore, a GA chromosome must be transformed so that it has to contain only the va-

lues from 0 to Max, where Max represents the number of different genes within the current array.

Suppose that we want to evolve the update strategy of a PSO algorithm with eight particles. This means that the $SwarmSize = 8$ and all chromosomes of the macro level algorithm will have eight genes whose values are in the $[0, 7]$ range. A GA individual with eight genes can be:

$$C_1 = (2, 0, 4, 1, 7, 5, 6, 3).$$

For computing the fitness of this chromosome it is used a swarm with eight individuals and, during iteration, the following updates are performed:

$$\begin{array}{l} update(Swarm[2]),\\ update(Swarm[0]),\\ update(Swarm[4]),\\ update(Swarm[1]),\\ update(Swarm[7]),\\ update(Swarm[5]),\\ update(Swarm[6]),\\ update(Swarm[3]). \end{array}$$

In this example all the eight particles have been updated once per iteration.

Let us consider another example which consists of a chromosome C_2 with 8 genes that contain only 5 different values.

$$C_2 = (6, 2, 1, 4, 7, 1, 6, 2)$$

In this case, particles 1, 2 and 6 are updated two times each and particles 0, 3 and 5 are not updated at all. Because of that it is necessary to remove the useless particles and to scale the genes of the GA chromosome to the set $\{0, \ldots 4\}$. The obtained chromosome is:

$$C_2' = (3, 1, 0, 2, 4, 0, 3, 1).$$

The quality for this chromosome will be computed using a swarm of size five (five swarm particles), performing the following eight updates:

$$\begin{array}{l} update(Swarm[3]),\\ update(Swarm[1]),\\ update(Swarm[0]),\\ update(Swarm[2]),\\ update(Swarm[4]),\\ update(Swarm[0]),\\ update(Swarm[3]),\\ update(Swarm[1]). \end{array}$$

Performing this transformation, we can obtain another swarm, which has a new size. The described model evolves only the update strategy for a PSO algorithm, but it can find the optimal swarm size in the same time, even if this parameter is not directly evolved.

We evolve an array of indexes based on the information taken from a function to be optimised. With other words, we evolve an array that contains the update order for the PSO algorithm. This algorithm is used for finding the optimal value(s) of a function. The quality of the update strategy is given by the performance of the PSO algorithm.

Note that the mentioned mechanism should not be based only on the index of the particles in the $Swarm$ array. This means that it would not be interested in updating a particular position since that position can contain (in one run) a very good individual and the same position could hold a very poor individual (during another run). For instance it is easy to see that all GA chromosomes, encoding permutations, perform similarly when averaged over (let's say) 1000 runs.

In order to avoid this problem the $Swarm$ array is sorted ascending (after each iteration) based on the fitness value. The first position will always hold the best particle at the beginning of iteration. The last particle in this array will always hold the worst particle found at the beginning of iteration. In this way it is known that $update(Swarm[0])$ will mean that the respective particle is not updated, but the best particle at the beginning of the current iteration will.

3.2 Fitness assignment

The model for evolving PSO update strategy is structured on two levels: a macro level and a micro level. The macro-level is a GA that evolves the update strategy of a PSO algorithm. For this purpose, a particular function is used as training problem. The micro level is a PSO algorithm used for computing the quality of a GA chromosome from the macro level.

The array of integers encoded into a GA chromosome represents the update order for the particles used by a PSO algorithm for solving a particular problem. The evolved order is embedded within a modified Particle Swarm Optimization algorithm as described in section 3.3.

Roughly speaking, the fitness of a GA individual is equal to the fitness of the best solution generated by the PSO algorithm encoded into that GA chromosome. But, since the PSO algorithm uses pseudo-random numbers, it is very likely that successive runs of the same algorithm will generate completely different solutions. This problem can be handled in a standard manner: the PSO algorithm encoded by the GA individual is run multiple times (50 runs in fact) and the fitness of the GA chromosome is averaged over all runs.

3.3 The algorithms

The algorithms used for evolving the PSO update strategy are described in this section. Because the hybrid technique from [4] combines a GA and a PSO algorithm within a two-level model, two algorithms are described: one for macro-level (GA) and another for micro-level (PSO algorithm).

3.3.1 The macro-level algorithm.

The macro level algorithm is a standard GA [7] used for evolving the update order of particles. We use steady-state evolutionary model as underlying mechanism for our GA implementation.

3.3.2 The micro-level algorithm

The micro level algorithm is a modified PSO algorithm [11] used for computing the fitness of a GA individual from the macro level.

The algorithm is quite different from the standard sychronous PSO algorithm [11] and from the asynchronous PSO algorithm [1, 15].

Standard PSO algorithm works on two stages: one stage that establishes the fitness, *pBest* and *nBest* values for each particle and another stage that determines the velocity and makes update for each particle. Standard PSO usually works with two populations/swarms. Individuals are updated by

computing the *pBest* and *nBest* value using the information from the previous population. The newly obtained individuals are added to the current population.

Asynchronous PSO algorithm works only in one stage: the fitness, *pBest*, *nBest*, velocities and positions for each particle are continuously updated. The particles are considered one by one for these modifications, following a preestablished order: the first particle, the second particle and so on.

The developed algorithm performs all operations in one stage only: determines the fitness, *pBest*, *nBest* and velocity values only when a particle is about to be updated. In this manner, the update of the current particle takes into account the previous updates in the current iteration. This PSO algorithm uses only one population/swarm. Each updated particle will automatically replace its parent. More, the genes of GA chromosome indicate the update order of the particles. The PSO individuals are not modified one by one following the initial order (1, 2, 3 ...), but following the sequence encoded into the GA chromosome.

In [4] were presented some numerical experiments for evolving the PSO update strategies. The obtained results have proved the effectiveness of this approached.

4. LESSONS LEARNT DURING THE EVOLUTION

We will try to identify some rules in the update strategy of the swarm during the evolving process. We want to identify these rules because they can help us to design better PSO algorithms. Before we detail our analysis, we will give a briefly definition for the swarm rule.

A swarm rule is a sequence of operations, which can be repeatedly used to generate new swarm or sub-swarms. The set of rules regards the order of updates (for instance this order could be correlated with particle quality: update firstly the best particles of the swarm and than, the weaker particles), the frequency of updates (some particles are updated more frequently than others particles are during the search process), the swarm size. These rules could emphasize some lessons learnt by particles during the evolutionary process.

All the numerical experiments performed in this paper are based on a PSO algorithm that optimizes the function $f(x) = \sum_{i=1}^{n} (i \cdot x_i^2)$, where $x_i \in [-10, 10]^n$ and $n = 5$.

4.1 Determining the optimal swarm size

First of all we analyzed the evolution of the swarm size along with the number of GA generations. In Figure 1 we depicted the evolution of the swarm size (swarm whose update strategy is encoded into the GA chromosome) for several particular GA generations.

We can observe that the smallest swarm from the first GA generation contains only four particles and the biggest swarm contains eight particles. These statistics are repeated during the next two generations. Starting with the fourth generation, the smallest swarm is composed by five particles and the biggest swarm by six particles. During the rest of generations, all swarms contain six particles, this size representing, probably, the optimal swarm size for the current problem.

In Figure 2 we depicted the evolution of the minim, maxim and average of the swarm size along with the number of GA generations. The average of swarm size in a particular generation is computed as the sum of swarms size (the swarms encoded into all GA chromosomes) over the number of individuals from the GA population. We can observe that the average of the swarm size decrease from 6.15 in first generation to 5.95 in the second generation and to 5.85 in the third generation. Starting with the next generation, the mean of the swarm size increases and it is stabilized at level six - this value seams to represent the optimal size of the swarm. Note that in the forth generation there are more swarms that contain only five particles than in the fifth generation and this fact determines the diminution of the mean swarm size.

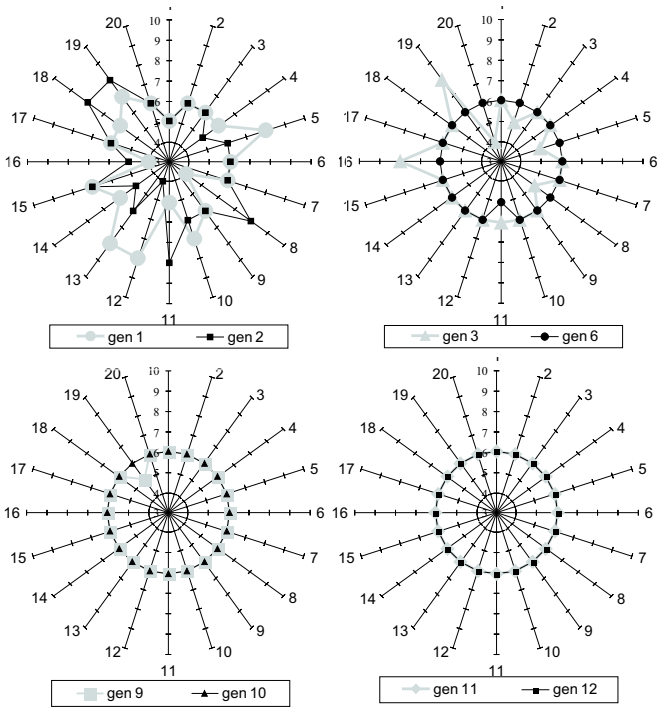

Figure 1: The evolution of the swarm size during the first two generations (top-left image), during the third and the sixth generations (top-right image), during the ninth and the tenth generations (bottom-left image) and during the eleventh and twelfth generations (bottom-right image). On each ray is depicted the size of a swarm whose update order is encoded into a GA chromosome (we have 20 rays because we work with a GA population composed by 20 chromosomes)

4.2 Which particles are updated more often?

The evolution of update frequencies for each particle along with the number of generations is presented in Figure 3. For computing this statistic, we calculated the average number of updates for each particle in all GA chromosomes. For instance, it is possible to update the best particle (which is the first particle from the swarm because they are sorted based on their quality) for two times in a GA chromosome and for three times in other GA chromosome. With other

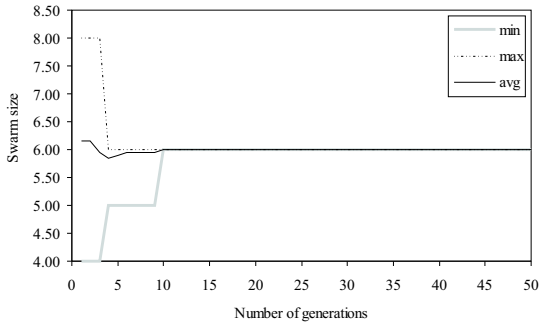

Figure 2: The evolution of the minim, maxim and average of the swarm size along with the number of generations.

words, the first particle is updated for an average of 2.5 times (supposing that we have only two GA chromosomes for this example). For obtaining a general synthesis during all the evolution process, this average is computed for each particle that can be in a swarm (in our case it is possible to have maxim ten particles) over all individuals from GA population (20 chromosome in this experiment).

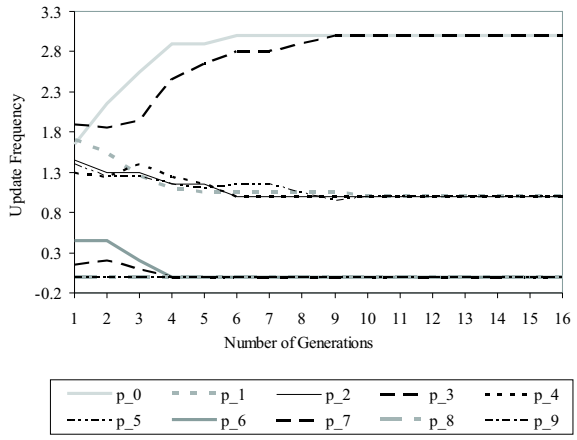

Figure 3: The evolution of the update frequency for each particle during the GA generations.

Taking into account the frequency of updates performed for a particle, we can observe in Figure 3 that most frequent updated particle is the first one. The best particle is updated for an average of 1.65 times in the first GA generation. This average rises up to three during the next five generations and rests at this level for the remaining generations. Note that in the first GA generation the second and the forth particles are updated more frequently than the first one, but taking into account the frequencies from all generations, the first particle is most updated.

Although the swarm particles are sorted based on the fitness, the next frequent updated particle is the forth particle. The average of update frequency for this particle starts from 1.9 times in first GA generation and increases also to three, but this value is obtained only in the ninth generation.

Unlike the first and the fourth particles, the rest of the particles (exception the ninth and the tenth particles) are updated fewer times as the generation number increases. Starting with the forth generation the seventh and the eighth particles are not updated. Particles three and five are updated only for one time starting with generation six and particles two and six are updated for one time starting with generation ten.

Another remark regards last two particles from the swarm: the ninth particle and the tenth particle (the "worst birds" of the swarm). These particles never are updated. With other words, a smaller swarm (only with eight particles) can solve our problem.

4.3 Analyzing the swarm update order

Several rules identified during the evolution of PSO update strategy are investigated in this section.

The qualities of the particles from the PSO algorithm whose update order is encoded into the best GA chromosome (into a particular generation) are presented in Figure 4.

The values on Ox scale indicate the update order for the particles. The Oy values indicate the quality of the particles: less high bars indicate better particles.

The first graphic (top-left-corner image) presents the initial swarm with six different particles. Firstly, the forth particle is initialized, then the second particle, the first particle, the fifth particle, the second particle once again, the third particle and so on.

In the second graphic (top-middle image) the sixth particle is updated at the ninth step and than, at the tenth step, the sixth particle is updated again. After this last modification, the quality of the sixth particle (given by the $pBest$ value) is improved. For instance, this situation can be observed also in the 26^{th} iteration depicted in the top-right-corner image. During the next iteration, firstly, the second particle is updated for two times (obtaining a quality improvement) and then the first particle is modified for two times (obtaining quality amelioration also in this case).

The quality of the second and the third particles is improved in the last two iterations.

Table 1 presents the number of updates performed for each swarm particle in a particular run of the proposed algorithm.

Table 1: The number of updates made for each swarm particle (P). Each column represents the moment (M) of update. Each row is reserved for a particle.

$P \mid M$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
p_0	0	10	83	11	0	11	52	19	2	32
p_1	22	50	2	1	69	33	42	5	0	3
p_2	21	1	9	19	3	52	0	35	26	22
p_3	64	17	13	18	26	5	0	13	49	12
p_4	2	16	3	53	8	2	0	27	11	16
p_5	1	13	0	2	2	4	8	9	22	22
p_6	0	2	0	3	2	2	5	1	0	3
p_7	0	1	0	2	0	1	1	1	0	0
p_8	0	0	0	1	0	0	1	0	0	0
p_9	0	0	0	0	0	0	0	0	0	0

As we already said, we performed ten updates inside a swarm even that the swarm size is smaller than ten (in this case some particles are updated for more times). For each particle is quantified the number of updates performed at each moment (because we performed ten evaluations for each swarm, obvious we have ten time moments). The particles are indexed based on the $pBest$ value.

The most frequent updated particle is the best particle (p_0) - it was updated for 83 times, but at first update moment the fourth particle (p_3 - a weaker particle) is modified. Note that, before the best swarm particle is modified, other two low-fitness particles are updated (p_1 and p_2) and after the modification of the best particle the worst particle is most updated.

Even that the best particle is updated for more times during PSO iteration, these best particle updates are not consecutively updated. Other updates (for weaker particles) are intercalated between these updates.

4.4 Summarizing the analysis

Analyzing the evolved update order for particles and the rules that appear during the evolution, we can conclude that:

- the proposed model is able to determine the optimal swarm size at the end of evolutionary process,
- the evolved update order is based on the particle quality - best particles are updated, in general, before updating the worst particles,
- frequency of updates - the best particles are more frequently updated than the worst particles; even that the best particle from a swarm is updated for more times, these updates are not consecutive. There are some updates, for other particles, performed between two successive updates for the best particle.

5. CONCLUSION AND FURTHER WORK

In this paper we have performed an analysis of the evolution of PSO algorithms. The model of the PSO algorithm involved in this analysis has several particular properties:

- it is different by the standard synchronous PSO were all particles are updated simultaneously,
- it is similar with the asynchronous PSO where the particles are updated continuously, but it is more general because it considers a dynamical update order and not a predefined one (as in asynchronous model). Moreover, the update order takes into account the particle quality.

Note that according to the No Free Lunch theorems [24] we cannot expect to design a perfect PSO which performs the best for all the optimization problems. This is why any claim about the generalization ability of the evolved PSO should be made only based on the results provided by numerical experiments.

In addition, we tried to analyze the data generated during the evolution of the update strategy. This will help us to understand the nature of the PSO algorithm and to design PSO algorithms that use larger swarms. Based on the analysis of the evolved update strategy, we can conclude several remarks. The first one is that the best particles from the swarm are updated most frequently. More than that, the evolution process can determine the optimal size for the swarm.

Further work will be focused on:

- several other kinds of function optimisation problem will be consider, to try to get more generic results, or to try to evolve PSO to solve a much more challenging, more interesting kind of problem,
- evolving more parameters of the PSO algorithm (independently, one by one, or synchronously),
- studying the generalization ability of the evolved PSO algorithm (how well it will perform on some new and difficult problems),
- designing an evolutionary algorithm able to identify by itself the rules analyzed in this paper and studying if these rules will actually improve the quality of a PSO algorithm in terms of convergence speed or accuracy of the solution.

6. REFERENCES

[1] A. Carlisle and G. Dozier. An off-the-shelf pso. In *Particle Swarm Optimization Workshop*, pages 1–6, 2001.

[2] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multi-dimensional complex space. *IEEE-TEVC*, 6:58–73, 2002.

[3] C. A. Coello Coello and M. Salazar Lechuga. MOPSO: A proposal for multiple objective particle swarm optimization. In *Congress on Evolutionary Computation CEC'2002*, volume 2, pages 1051–1056. IEEE, 2002.

[4] L. Diosan and M. Oltean. Evolving the structure of the particle swarm optimization algorithms. In *European Conference on Evolutionary Computation in Combinatorial Optimization EcoCOP2006*, volume 3906 of *LNCS*, pages 25–36, 2006.

[5] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. on Systems, Man, and Cybernetics–Part B*, 26(1):29–41, 1996.

[6] R. C. Eberhart and Y. Shi. Particle swarm optimization: Developments, applications, and resources. In *Congress on Evolutionary Computation CEC'2001*, pages 81–86. IEEE, 2001.

[7] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison Wesley, 1989.

[8] T. Hendtlass. A combined swarm differential evolution algorithm for optimization problems. In L. Monostori, J. Váncza, and M. Ali, editors, *Proceedings of the 14th IEA/AIE 2001*, volume 2070 of *LNCS*, pages 11–18, 2001.

[9] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[10] X. Hu and R. Eberhart. Multi-objective optimization using dynamic neighborhood particle swarm

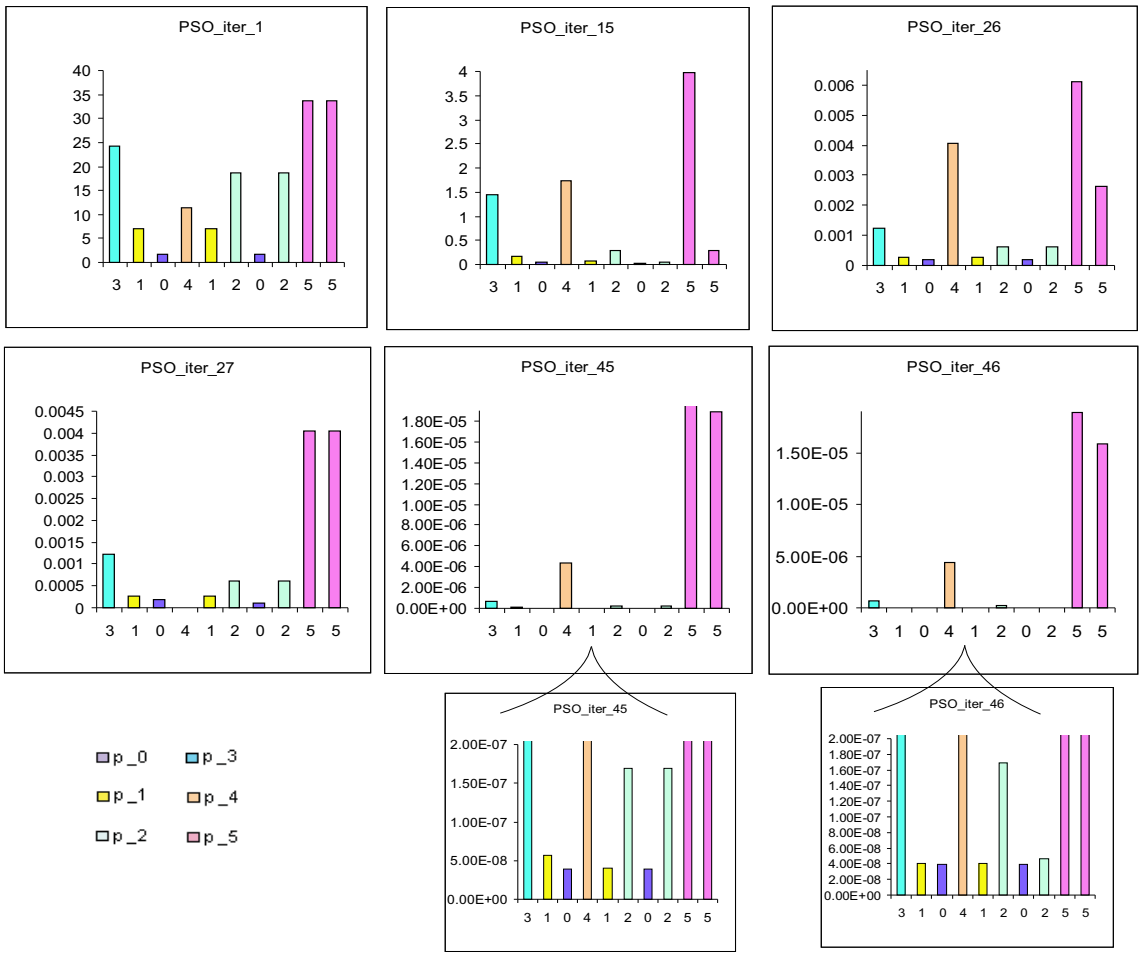

Figure 4: The updates order encoded into the best GA chromosome from one generation of the GA. We depicted different iterations of the PSO algorithm that updates the particles based on the order encoded into GA chromosome. Different particles are represented with different colours and the height of the columns represents the quality of a particle - less higher, better. Note that for a particular PSO iteration only the swarm state at the end of that iteration is depicted here - even that the updates are performed gradually, taking into account the previous modifications performed into the current PSO population.

optimization. In *Congress on Evolutionary Computation CEC'2002*, volume 2, pages 1677–1681. IEEE, 2002.

[11] X. Hu, Y. Shi, and R. Eberhart. Recent advances in particle swarm. In *Congress on Evolutionary Computation CEC'2004*, volume 1, pages 90–97. IEEE, 2004.

[12] J. Kennedy. The behavior of particles. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII, 7th International Conference, EP98*, volume 1447 of *LNCS*, pages 581–589, 1998.

[13] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE, 1995.

[14] J. Kennedy and R. C. Eberhart. The particle swarm: Social adaptation in information-processing systems. In D. Corne, M. Dorigo, and F. Glaover, editors, *New Ideas in Optimization*, pages 379–387. McGraw-Hill, 1999.

[15] B. Koh, A. George, R. Haftka, and B. Fregly. Parallel asynchronous particle swarm optimization. *International Journal for Numerical Methods in Engineering*, 67(4):578–595, 2006.

[16] H. Kwong and C. Jacob. Evolutionary exploration of dynamic swarm behaviour. In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC'2003*, pages 367–374. IEEE, 2003.

[17] W. B. Langdon, R. Poli, and C. R. Stephens. Kernel methods for PSOs. Technical report, Computer Science, University of Essex, UK, 2005.

[18] C. K. Mohan and B. Al-kazemi. Discrete particle swarm optimization. In *Workshop on Particle Swarm Optimization 2001*, 2001.

[19] E. Ozcan and C. Mohan. Particle swarm optimization: surfing the waves. In *Congress on Evolutionary Computation CEC'1999*, pages 1939–1944. IEEE, 1999.

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

[21] R. Poli, W. B. Langdon, and O. Holland. Extending particle swarm optimisation via genetic programming. volume 3447 of *LNCS*, pages 291–300, 2005.

[22] Y. Shi and R. C. Eberhart. Empirical study of particle swarm optimization. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Congress on Evolutionary Computation CEC'1999*, volume 3, pages 1945–1950. IEEE, 1999.

[23] F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, South Africa, 2002.

[24] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE-TEVC*, 1(1):67–82, 1997.

[25] A. E. M. Zavala, A. H. Aguirre, and E. R. V. Diharce. Particle evolutionary swarm optimization algorithm (PESO). In *ENC*, pages 282–289. IEEE Computer Society, 2005.

[26] W. Zhang and X. Xie. Depso: hybrid particle swarm with differential evolution operator. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics,*, pages 3816–3821. IEEE, 2003.