POSC/EEA-ESE/60980/2004

EPSO – Evolutionary Particle Swarm Optimization, a new meta-heuristic with applications in power systems

September 2006

REPORT ON

EPSO – theory overview

Prepared by: Hrvoje Keko
Prepared for: EPSO project

Abstract

EPSO – Evolutionary Particle Swarm Optimization [1] algorithms are evolutionary methods that borrow the movement rules from Particle Swarm Optimization (PSO) methods and use it as a recombination operator that evolves under pressure of selection. This hybrid approach builds an algorithm that, when applied to complex problems in power systems, has already proven to be more efficient, accurate and robust than classical evolutionary methods or classical PSO. Even though EPSO algorithm has already proven its efficiency on the practical problems, winning the competition with other metaheuristic methods, it can still be improved and made even more efficient.
Investigating the algorithm solely by extensive empirical testing could lead to improvements of its performance on particular problem sets. However, without some theoretical understanding of the algorithm, a general improvement of algorithm's performance is rarely possible.
As the *No Free Lunch* theorem [2] by Wolpert and Macready states, there is no such algorithm that is generally, on average, superior to any other competitor, but purely testing and demonstrating algorithm's superiority on a given problem does not provide understanding on *why* the algorithm actually works and how to improve the algorithm's performance. The usefulness of theory in the domain of evolutionary computation is often questioned, but "theory" should be understood not only as collection of mathematical theorems and proofs, instead it includes a collection of experience or knowledge useful for making predictions of the evolutionary system performance. The theoretical approach should clearly define the domain of applicability of the algorithm, and if possible, present the convergence proofs and efficiency results. Theoretic investigation of evolutionary methods is, however, difficult and, in strict mathematic terms, often impossible with complicated real-life problems. In other words, convergence can often be strictly mathematically proven only on very simple or special (artificial) cases. Even with these limitations taken into account, theoretical investigation of evolutionary algorithms has

Evolutionary Computation

Brief history of evolutionary computation

Even though the exact term *evolutionary computation* hasn't been coined until the 1990s, the basic ideas of evolutionary computation date to the 1950s. The first works related to using an evolutionary process for computer problem solving deals with the use of an evolutionary algorithm for automatic programming – finding a program that calculates a given input-output function.

By the mid-1960s, visionary researchers have developed "classic" methods of evolutionary computation and the bases for today's three main forms of evolutionary algorithms have been clearly established. The roots of evolution strategies (ES) have been established in Technical University of Berlin by three students, P. Bienert, I. Rechenberg and H.P. Schwefel. At the same time, in San Diego, California, Lawrence Fogel was dealing with the first instances of evolutionary programming (EP). At the University of Michigan in Ann Arbor, the works of John Holland led to development of genetic algorithms (GA).

The basic idea behind evolutionary computation was to employ natural evolution as a powerful problem-solving paradigm. All evolutionary algorithms utilize the collective learning process of a population of individuals. Usually, each individual represents a search point in the space of potential solutions. Put simply, evolutionary computation methods evolve solutions to problems and try to reach the optimum solution. All evolutionary methods rely on *fitness function* – a specific function that provides a measure of individual's quality, and thereby provides the ranking of solutions. Fitness function is an essential ingredient of every evolutionary algorithm, as a process for ranking the individuals and assigning the measure of quality to each individual.

While GA, ES and EP all share the same algorithmic approach where individuals are not confronted with one another but measured against an external common selection function, genetic algorithms differ significantly from the other two approaches. The genetic algorithms focus on genetic material – genotype, not phenotype – phenotypic (outer, visible) characteristics of the individuals. This means that the operators in genetic algorithm rely on discrete genetic representation of individuals. They deal with alternating and exchanging of genes between the individuals. Finding the optimum solution to a particular problem is 'translated' into a problem in finding the genes that build the optimal individual [1], but each individual is evaluated on the base of its phenotypic characteristics.

The ES and EP approach is different. There is no real gene level in evolutionary strategies and evolutionary programming; they do not separate the worlds where a variant of the solution is generated from the one where the selection acts, like in genetic algorithms. Moreover, ES and EP aren't two distant paradigms; instead they could be viewed as variations on a same theme. Actually, it's the geographical independence in development of EP and ES that led to creation of subsequent schools of followers. After years of research and applications in both directions, they can be attributed with the same generic name like *phenotypic methods*,

attributing their dependence on phenotypic characteristics of the individuals in the population.
After the 1960s, over the following 25 years all these branches developed quite independently of each other. The international workshop entitled *Parallel Problem Solving from Nature*, held in Dortmund in 1991, presented an organized effort for providing a forum of interaction between these research communities. Subsequently, in 1993, the term *evolutionary computation* was coined as a general term covering all major forms of evolutionary computation. Since EPSO has more in common with the ES and EP approach than with GA approach, ES/EP theory will be presented.

Evolution Strategies

Introduction

In 1965, three students of Technical University of Berlin, P. Bienert, I. Rechenberg and H.P. Schwefel did not aim at devising new kind of optimization procedure. Working at Institute of Fluid Mechanics, they wanted to construct a research robot that should perform a series of experiments on a flexible, slender three-dimensional body in a wind tunnel, in order to minimize its drag. However, when tested on joint plate, the two-dimensional demonstration facility, the classical methods, optimizing one variable at a time or using gradient techniques, failed and became prematurely stuck in local optimum. Rechenberg hit the idea to use dice for random decisions, and June 12^{th}, 1964 was the birth date of the first version of an evolutionary strategy, later called the (1+1) ES. The first evolutionary strategy had discrete binomially distributed mutations centered at the ancestor's position and only one parent and one descendant per generation.
The ES was tested on a mechanical calculating machine by Schwefel before it was used to optimize the joint plate. Later, Bienert constructed a kind o robot that could perform the optimizing actions and decisions automatically.
First computer experiments and analytical investigations were performed by Schwefel in 1965, and the first problems with the new method indicated that the strategy can become stuck prematurely, at solutions not even locally optimal. This led to using normally instead of binomially distributed mutations, and also to later theoretical investigations of ES. First Rechenberg, and then Schwefel analyzed and improved their ES. Rechenberg's work in 1971 introduced convergence rate theory for optimizing a function with $n \gg 1$ variables. This work proves that the convergence velocity (distance traveled into useful direction per algorithm's iteration) is inversely proportional to number of variables and proportional to orographic measure of isohypses' curvedness of objective function. Moreover, he has proven that the linear convergence order is achievable if the mutation strength in the algorithm is adjusted to proper order of magnitude. Finally, his work shows that the optimal mutation strength corresponds to a certain success probability independent of dimension of the search space. The optimal success probability provided on two model functions was in range of 1/5. Rechenberg's work also introduced the first type of multimembered ES, the $(\mu+1)$ ES

with μ parents, two of which are chosen to produce a descendant. The selection is then the extinction of the worst individual.

H.P. Schwefel in 1974 introduced two main forms of multimembered evolution strategies:

- $(\mu+\lambda)$-ES, where there is $\lambda \geq 1$ descendants, and to keep the population size constant, in each generation λ worst individuals are discarded
- (μ,λ)-ES, in which selection takes place among the λ offspring only, and the parents are forgotten. Obviously, in this case $\lambda \geq \mu$ must hold.

Like any classical method, the performance of evolution strategies depends on the adjustment of internal parameters. The multimembered (μ,λ) and $(\mu+\lambda)$ evolution strategies became more popular due to their capability of self-adaptation, they arose from otherwise ineffective incorporation of *mutatable mutation parameters*. Schwefel's later research work continued in the direction of theoretical investigation of ES. In the 1980s, the first notions of *self-adaptation by collective learning* came up, but the theoretical background was formed in late 1990s, in the works of H.G. Beyer and books by Schwefel, Rechenberg and Bäck [4],[5],[6],[7],[8]. Many other results have been produced in the German ES community, consisting of the group at Berlin (Rechenberg, from 1972) and the group in Dortmund (Schwefel, from 1985). Even though there is still much work to be done in order to establish on solid grounds a general theory of generalized evolution strategies, the significant theoretical results available until today will be explained in the following sections.

ES algorithm fundamentals

This section presents the ingredients and the notation for contemporary ES, capable of self adaptation. The usual goal of an evolution strategy is to optimize given objective (quality) function $\mathrm{F}(\mathbf{y})$ with respect to a set of decision variables or control parameters $\mathbf{y}=(y_1,y_2,\ldots)$ called *object* parameters. Search space can be any set of data structures of finite but not necessarily fixed length. Examples include real-valued N-dimensional search space, integer search space, binary search space, space of permutations etc. Evolution strategies operate on populations B of individuals α; an individual α_k not only comprises the specific vector of object parameters $\mathbf{y}_k$ and its objective function value $\mathrm{F}_k=\mathrm{F}(\mathbf{y}_k)$, but also a set of *endogenous strategy parameters* $\mathbf{s}_k$. The endogenous strategy parameters regulate the ES and can evolve during evolution process. Also they are a prerequisite for a self-adaptive ES.

Though specialized versions of ES do exist, in theoretical research usually focuses on a canonical set of ES features. In the beginning there were two different forms of ES: $(\mu+\lambda)$ and (μ,λ)-ES, differing in the principle of selection, as noted in the previous chapter. In 1995 [4], a broader definition of $(\mu,\kappa,\lambda,\rho)$ was proposed.

- μ - number of parents in a generation
- κ - number of generations (reproduction cycles) that an individual can survive
- λ - number of offspring created in one generation
- ρ - number of parents (direct ancestors) for each individual

Within one ES evolution step, λ offspring individuals are generated from the set of μ parent individuals. Each new individual is generated from ρ parents involved in procreation. These ρ parents recombine to create one new individual. The special case $\rho = 1$ means there is no recombination; instead a new individual is a clone of his only parent. Each new individual is created by randomly choosing ρ parents from the parent pool, and then the new individual is built by recombination. After that, the new individual's strategy parameters and object parameters are mutated. The process is repeated λ times, then selection takes place. Depending on whether or not parents can survive, μ best individuals are chosen either from the offspring exclusively (the *comma* selection) or from both parents and the offspring (the *plus* selection). The remaining parameter limits the parents' survival period to κ generations, i.e. the individual can survive only up to κ generations. Finally, after each generation cycle is finished, a termination condition is checked. Termination condition can be based on resource criteria (given number of generation or CPU time) or algorithm convergence criteria (either in the space of fitness values, object parameters or strategy parameters).

The $(\mu, \kappa, \lambda, \rho)$-notation doesn't include all the possible parameters – contemporary ES have more parameters, for instance: P- the starting population, mutation operator, mutation probability, recombination operator, recombination probability, selection operator and others, as well, some of them associated with chosen operators.

Favorable properties of ES operators

Each evolutionary algorithm needs a goal oriented selection operator, in order to guide the search into promising regions of the object parameter space. Selection is an antagonist of the variation operators, mutation and recombination, whose primary role is exploration of the search space. Selection creates a new parental population by a process which aims to drive fitter individuals into the selection pool for the next algorithm generation. However, obviously, solely these properties of selection operator do not guarantee algorithm's convergence, they are necessary but not sufficient properties of a working ES.

Mutation operator, unlike selection operator, is a variation operator in the ES. Even though there is no established design methodology for the operator, Beyer [9] has proposed three general guidelines that a mutation operator should obey:

- reachability – given a parental state, this ensures that any other state can be reached within a finite number of mutation steps. This is a prerequisite for global convergence.
- unbiasedness – while selection exploits the fitness information in order to guide algorithm into promising regions of the search space, mutation should not introduce any bias, and being maximally unbiased is a design criterion for variation operators. A

maximum entropy principle follows naturally, and for real valued search spaces $\mathbb{R}^N$ Gaussian normal distribution proves to be a good choice.

- scalability – states that the mutation strength, or an average length of mutation step should be tunable in order to adapt to properties of fitness landscape. Since the properties of the fitness landscape are determined by objective function *and* the variation operators, it can be expressed as the causality concept, presented by Rechenberg in and formalized by Sendhoff et al in 1997 [10]. This concept states that on average small changes in the genetic level should result on average in small changes in the fitness values.

It must be noted that even the operators that do *not* comply with these rules can be successful in a specific application.

Recombination, unlike mutation, uses information from up to ρ parents. Recombination stands as a word that designates a number of distinct procedures that share the property of building an individual departing from a set of parents. There are two classes of recombination operators in ES – dominant recombination and intermediate recombination. The dominant recombination coordinate-wise randomly selects elements from the parents and copies the value of element from the parent chosen as dominant for that component. In other words, the *k*-th component of the descendant is determined solely by *k*-th component of the dominant parent. This scheme is also called discrete recombination or uniform crossover. Another variant of the scheme is multipoint crossover scheme, usually used in genetic algorithms, where one selects a number of crossover points, and offspring successively receives a part of genetic material from parents.

The intermediate (intermediary) recombination, on the other hand, takes all ρ parents into account and creates offspring as center of mass (centroid) of parent vectors.

Building block hypothesis stated by Goldberg [11] in 1989, commonly used as a supporting theory for building recombination operators in GA community, explains the use of recombination as an operator preserving good *building blocks* from the parents and combining good properties of parents to create good offspring. While the explanation looks intuitively appealing, finding the test function to theoretically and practically support the hypothesis was very difficult. Over the 1990s, ES theory research instead revealed another mechanism of recombination – the genetic repair effect [12] and corresponding genetic repair hypothesis. The GR hypothesis stands on the opposite standpoint to BBH hypothesis – it is not the *different* (i.e. desired) features that recombination operator transfers to the offspring; instead it's the *common* features, similarities from the parents. It is reasonable to assume that corresponding components in parents that are similar to each other carry a higher probability of being beneficial with respect to individual's fitness. The best a variation operator can do is to preserve these common features, since other might be irrelevant or even harmful [13]. By conserving the beneficial parts, the recombination operator damps the harmful effect of mutation causing the decrease in fitness. Genetic repair theory is easily explained on real-valued search spaces, while its derivation on combinatorial search spaces still remains a challenge.

Theoretical performance measures: quality gain and progress rate

There is a long tradition of theoretical research in the ES community, dating back to its beginnings in the 1960s. All theoretical results in the realm of ES have to rely upon simplifications of the situation investigated. The most important goal of mathematical theory in ES is predicting its dynamic behavior, mathematically describing and investigating the ES performance in time domain. ES are Markovian processes and are governed by Chapman-Kolmogorov equations – the generalization of Markov chains for discrete search spaces. These equations provide a vast amount of information, and theoretical research is more directed towards analyzing the aggregated information related to optimization performance of ES, like:

- the dynamics of the average, best and worst fitness
- dynamics of expected distance towards the optimum
- local performance properties
- global performance properties (convergence proofs, orders of convergence, running times and time complexities)

One of local performance properties useful in theoretical examination is called *success probability* P_s – it is the probability of improving an individual by mutating it. For the (1+1) ES, this is exactly the same probability for the descendant replacing a parent. Assume that the fitness function $\mathrm{F}(\mathbf{y})$ can be locally approximated by the *local quality function* $\mathrm{Q}(\mathbf{x})$, or in other words the transition from parental state y_k at generation *k* to generation *k+1* by mutation **x** induces a fitness change expressed by local quality function.

$$\mathrm{F}(\mathbf{y}_{k+1}) - \mathrm{F}(\mathbf{y}_k) = \mathrm{F}(\mathbf{y}_k + \mathbf{x}) - \mathrm{F}(\mathbf{y}_k) =: Q(\mathbf{x})$$

The performance of the ES can then be described by *quality gain* [7]. The quality gain is defined as the expectation of the fitness change induced by mutation **x**.

$$\langle Q \rangle := E\{Q\}$$

Another way of describing ES performance is *progress rate* φ. If the optimum of $\mathrm{F}(\mathbf{y})$ is located at $\hat{\mathbf{y}}$, then the progress rate is the expectation of distance traveled from generation *k* to generation *k+1*:

$$\varphi := E\left\{\left\|\hat{\mathbf{y}} - \mathbf{y}_k\right\| - \left\|\hat{\mathbf{y}} - \mathbf{y}_{k+1}\right\|\right\}$$

From the theoretical point of view, describing ES performance in object parameter space, through progress rate, should be the preferred performance measure. Unfortunately, determination of progress rate is difficult and tractable for simple fitness functions only. This is one of reasons that sphere fitness function has a particular importance for investigating the ES:

$$\mathrm{F}(\mathbf{y}) = C\left\|\mathbf{y}\right\|^{\alpha},\ \mathbf{y} \in \mathbb{R}^n$$

All vectors lying on a particular hypersphere have the same value of sphere function, which makes this case symmetric and dependant only on one dimension – the hypersphere radius (or the norm of vector **y**). That is, the radius of the hypersphere can be used as an

aggregated state variable, reducing an analysis of algorithm performance from *N*-dimensional space to one-dimensional space.

Self-adaptation

The necessity of controlling endogenous strategy parameters became evident when running a simple, (1+1) ES with isotropic Gaussian mutations and constant mutation strength, on a sphere function. Even though it can be shown that (1+1) strategy is globally convergent, in practice after a period of improvements the strategy becomes extremely slow, because the system has lost its evolvability, due to fixed mutation strength. (1+1) case is particularly useful since it only relies on mutation which makes it easier to understand. An intuitive explanation how mutation strength influences performance of (1+1) ES follows from symmetric nature of mutations and smoothness of fitness landscape. If the mutation strength σ is chosen very small, on average the success probability will be ½, on average every second mutation will "hit" inside the hypersphere, and small mutations will have a high degree of success. However, mutation steps are scaled with mutation strength, progress towards the optimum is also scaled with σ.

$\sigma \to 0 : P_S \to 1/2, \varphi \to 0$

The performance will therefore be rather poor. The opposite case with σ very high makes the mutations produced from parental state too large and the success probability (probability of "hitting" inside hypersphere) will be very small.

$\sigma \to \infty : P_S \to 0, \varphi \to 0$

Between these two extremes there is a bandwidth of mutation strengths guaranteeing nearly optimal performance, called *evolution window* by Rechenberg.

Since both performance of the ES (progress rate) and success probability depend on mutation strength, a σ-control rule can be established. For the sphere function, optimal success probability that gives the maximum progress rate is $P_S \approx 0.27$, and for other functions it was close to 0.2. A compromise proposed by Rechenberg is the famous 1/5th rule: in order to obtain nearly optimal local performance of (1+1) ES in real valued search spaces, tune the mutation strength so that the measured success probability is around 1/5. According to Rechenberg one should evaluate the achieved progress rate, and if it is larger than 1/5, make $\sigma = a \cdot \sigma$, if it is smaller than 1/5, make $\sigma = q/\sigma$, where $a \in [1, 2]$.

This control heuristics represents the very simplest version of *deterministic* nonlocal adaptation. It became important since, for some problems, it represents a locally good approximation of given fitness landscape. For other problems, however, the good value of progress rate depends on the topology of search space. Rechenberg's control heuristics uses only the global information on success probability, it is restricted to application of only one strategy parameter, the fitness landscape also must obey certain properties and it works only for (1+1) ES. Therefore, a more flexible and *evolutionary* technique for adaptation was introduced: *self-adaptation* or σ-SA as called by Schwefel [8]. The principal idea rests on *individual* coupling of endogenous strategy parameters with the object parameters, so that each individual has its own set of strategy parameters. The parameters undergo

To obey scalability rule, mutation of the strategy parameters in real-valued search spaces must be performed multiplicatively, $\sigma_{k+1} = \sigma_k \cdot \xi$ where ξ is a random number sampled from a random generator. The expectance of ξ should be close to 1 and there are several random number distributions that have found a practical use:

- lognormal distribution, proposed by Schwefel, where mutation has the same probability of being doubled or divided in half $p_\sigma(\xi) = \frac{1}{\tau\sqrt{2\pi}}\frac{1}{\xi}\mathrm{e}^{-\frac{1}{2}\left(\frac{\ln\xi}{\tau}\right)^2}$
- exponential transformation of Gauss normal distribution, $\xi = \mathrm{e}^{\tau\,\mathrm{N}(0,1)}$
- symmetrical two-point distribution, $\sigma_{k+1} = \sigma_k \cdot (1+\beta)$ if $\mathrm{U}(0,1) \le 0.5$ and $\sigma_{k+1} = \sigma_k / (1+\beta)$ if $\mathrm{U}(0,1) > 0.5$ where U(0,1) is a random number sampled from uniform distribution

These schemes, and consequently the ES performance, depend on the value of *learning parameter*, τ in first two cases and β in third case. The learning parameter conditions the speed and accuracy of the σ-SA evolution strategy. Schwefel's theoretical results provide that the learning parameter should be proportional to $1/\sqrt{N}$ where *N* is the dimensionality of the problem. Beyer [14] proposed some practical formulas for choosing the learning parameter.

Once the initial transient phase of the algorithm is finished, optimal choice of the learning parameter leads the algorithm towards exhibiting a *linear convergence order*.

To achieve acceptable progress rates, a mechanism capable of reducing fluctuations is needed. According to genetic repair principle, recombination extracts similarities from the individuals – this is why recombination is recommended and has positive effect in self-adaptive ES [15], [16], [17].

Self-adaptation on the level of individuals, however, can also fail. The reason for such failures is "opportunism" – evolution rewards short term success, and if the local progress and global progress aren't positively correlated, evolution might choose offspring states leading towards local optimum. An example of approach for avoiding these premature convergence problems is comparing partially isolated evolution processes which lead to nested evolution strategies or meta-ES. Meta-ES, nested ES or hierarchically organized ES are strategies described as $\left[\mu'/\rho'+,\lambda'\left(\mu/\lambda+,\lambda\right)^{\gamma}\right]-\mathrm{ES}$: there are μ' parental populations of $(\mu/\lambda+,\lambda)$-ES that run without any communication for *γ* generations, where *γ* is an *isolation*

parameter. After *γ* generations, "outer" ES selection determines the best μ' of $(\mu/\lambda+,\lambda)$ strategies. There are several distinctive application approaches to nested ES:

- as meta-heuristic strategies for global optimization in simple search spaces, where outer ES performs global optimization and inner ES local search
- in search spaces composed from different kinds of simple search spaces, for example mixed-integer search spaces, neural networks design – in such application evolution of the structure is performed by outer ES and weights are evolved by inner ES
- solely optimizing the performance of inner ES through breeding by outer ES

Another approach to self-adaptation is a class of *deterministic* nonlocal techniques for adapting the mutation operators, called *cumulative path-length control* (Ostermeier, [18]). It relies on evolution path – in the simplest case, vector sum **v** of actually realized evolution steps over a number of last *G* generations. A mutation step is kept constant over *G* generations, so an expected length of **v** can be calculated.
If the chosen mutation step is too small, then selection will prefer larger steps and actual **v** will be larger than expected value. On the opposite, if the mutation steps are too large, smaller steps will be selectively preferred and **v** will be smaller than the expected value.
The basic idea is reminiscent of the $1/5^{th}$ rule, but there is a difference on how the nonlocal population information is used. The $1/5^{th}$ rule discards any search space information. More advanced implementations of cumulative path-length control introduce cumulative step-size control and adaptation of covariance matrices needed to produce correlated mutations.
After having experiments with global evolving mutation strength, and defining a global learning factor, the next step was decoupling the mutation strengths so that distinct variables undergo distinct evolution processes (i.e. setting *n* different mutation strengths for *n* objective variables defining an individual).
The sphere function model might be a good local approximation for many cases, but it assumes an isotropic topology of the search space. Some search spaces, however, are not isotropic and allowing distinct mutation strengths allows decoupling of the mutation rate according to distinct coordinate directions. A scheme with different mutation strengths allows decoupling of mutation rates according to the axial directions of search space.
All of these approaches establish different mutation rates while keeping them non-correlated. Even this is not enough for achieving effective performance in some search space – a correlation, dependence between evolution along some direction and some other direction should be established to assure the algorithm's performance.
The scheme with isotropic spaces assumes that length of a vector is given by $\|\mathbf{R}\| = \mathbf{R}^t\mathbf{R}$ and decoupling of variable mutation strengths is equivalent to assuming a diagonal metrics in the search space, i.e. $\|\mathbf{R}\| = \mathbf{R}^t\mathbf{D}\mathbf{R}$ where **D** is a diagonal matrix. Recognizing this, ES incorporated correlation between mutations as strategic variables, which is equivalent to constructing Mahalanobis metric in space, and length of a vector **R** is then given by $\|\mathbf{R}\| = \mathbf{R}^t\mathbf{T}\mathbf{R}$ where **T** is a full matrix. In ES, a formal mathematical representation of the

possible covariances of mutation distribution is adopted, with the basic concept of introducing new strategic variable - inclination angle α.
Given an inclination angle between two coordinate directions p and q, covariance matrix between these directions can be defined by transformation matrix

$$\mathbf{T}_{pq}(\alpha) = \begin{bmatrix} 1 & 0 & & & & & & & \\ 0 & 1 & & & & & & & \\ & & \ddots & & & & & & \\ & & & \cos\alpha_j & & & -\sin\alpha_j & & \\ & & & & 1 & 0 & & & \\ & & & & & \ddots & & & \\ & & & & 0 & 1 & & & \\ & & & \sin\alpha_j & & & \sin\alpha_j & & \\ 0 & 0 & & & & & & 1 & 0 \\ & & & & & & & & \ddots \\ 0 & 0 & & & & & & 0 & 1 \end{bmatrix}$$

where only lines and columns p and q have elements distinct from 0 or 1. The product of all $\mathbf{T}_{pq}$ matrices gives the covariance matrix **C**.
Without covariance matrix, adding a mutation to an individual in real-valued search space means adding a random vector with non-correlated components. Covariance matrix multiplies that vector and gives a random vector with normally distributed *and* correlated components. The strategic variables that establish these correlations are the inclination angles *α*. Setting all of these to zero makes the covariance matrix an identity matrix so the mutations will adapt independently. According to Schwefel, the angles in covariance matrix should be mutated according to normal distribution $\alpha_{k+1} = \alpha_k + z_k$, $z_k \in \mathrm{N}\left(0, \beta^2\right)$, and values of *β* are chosen according to experiments.
Aforementioned observations are based on single mutation operator, but it is not uncommon for ES to include several different mutation operators, ruled by probabilities.

Constraint handling

Handling feasibility constraints is an important issue for real-life usage of the optimization algorithm. Phenotypic nature of the ES enables constraints handling in a very natural way. During mutation phase, individuals can be checked for feasibility and discarded if they don't conform to constraints, and a replacement offspring is being generated until a feasible one is found. Another approach is to create mutation operators so that infeasible descendant cannot be created. This, however, might be a time-consuming solution so another approach is to penalize unfit individuals, lowering their level of fitness so that selection effectively eliminates these individuals. This extends to all types of algorithms relying on fitness measure and selection.

Evolutionary Programming

Introduction

Evolutionary programming was devised by Lawrence J. Fogel in 1960. At the time, artificial intelligence was mainly concentrated around heuristics and simulation of primitive neural networks. Fogel viewed these approaches as limited, since they model humans instead of the evolution process. L. Fogel perceived the evolution process as the process that produces creatures of increasing intellect, and intelligence to be based on adapting behavior to meet goals in a range of environments. Prediction was viewed as a key ingredient to intelligent behavior. Based on this, L. Fogel conducted a series of experiments on the use of simulated evolution of finite state machines to forecast nonstationary time series, whilst respecting arbitrary criteria. These experiments were published in a series of publication during the 1960s.

The evolutionary problem was defined as evolving an algorithm (a program) that would operate on sequence of symbols and produce an output symbol that maximizes the algorithm's performance in light of the next symbol to appear in the environment and a well defined payoff function. A population of finite state machines is exposed to the environment and each output symbol is compared with the next input symbol, and then the worth of prediction is measured with respect to the payoff function. Average payoff per symbol indicates the fitness of the machine. Offspring machines are created by random mutations of the parent machines, and the machines that had the greatest payoff became parent machines in the next generation.

The initial efforts of L.J. Fogel indicate some of early attempts to use simulated evolution to perform prediction, include variable-length encodings of population members, use representations that can take a form of sequence of instructions (a program), incorporate a population of candidate solutions (more than one) and coevolve evolutionary programs.

The general procedure of EP was successfully applied to problems in prediction, identification and automatic control. In mid-1980s the general EP procedure was extended to alternative representatives, beside finite state machines, including ordered lists for traveling salesman problem and real-valued vectors for continuous function optimization. In turn, this led to other applications in route planning, optimal subset selection and training neural networks. The internal self-adaptation of mutation variances of greatest importance for evolutionary programming's efficiency and it was investigated in works by D. Fogel throughout the 1990s. Over the 1990s the contacts were made between EP community and ES community and later the works in EP have diversified in even more directions.

Distinction and similarities between ES and EP

Like ES, EP is a phenotypic method [22]; it relies on phenotypic description of an individual instead of its genetic representation. David Fogel's article [19] in particular emphasizes that the evolutionary programming stresses behavioral change at the level of species, whereas the evolution strategies stress behavioral changes at the phenotypic level of single

$\mu + \lambda)$

The most exaggerated traditional difference between evolution strategies and evolutionary programming is related to the form of selection. While ES traditionally included elitist selection, where the best individuals at each generation is preserved and selected into the next generation, the EP traditionally preferred selection by stochastic tournament. The simplest version of stochastic tournament selection T(1,2) is the one that randomly samples two individuals from the population, and with externally fixed probability, selects the one with better fitness for the next generation. This is repeated until the required number of offspring is generated. Other kinds of tournament selections exist, like T(*m*,*n*) where the best *m* out of *n* individuals are selected. The first tournament method introduced in EP was slightly different: one solution competes with other solutions in pair-wise comparisons, like in sports tournaments, and a win is awarded to better solution or awarded probabilistically according to chosen formula. After that, the solutions with most wins become parents in the following generation.

Some models of EP, however, abandoned the stochastic nature of selection and turned to pure elitist processes, simply selecting λ best individuals to form the following generation. There are, of course, also ES that use stochastic tournament selection and EP with elitist selection. The EP-community also developed self-adaptive strategy, which can be related to the one emerged from ES. Evolution of the mutation strength parameter is governed by rule $\sigma_{g+1} = \xi\sigma_g$ where ξ is a random number given by $\xi = 1 + \tau\,\mathrm{N}(0,1)$ and τ is externally fixed learning parameter. This relates to the evolution in ES – lognormal operator, when expanded to linear term gives precisely that.

Perhaps the biggest distinction between the ES and EP relates to the early days of EP and its application to evolving machines which in turn led to specialization of EP in mutation variants, techniques and operators. This lead to fruitful improvements in the areas related to machine learning and classifier systems. An example of evolutionary programming with self adaptation is presented in [21] and [22] – the basic theoretical background of self-adaptation features of the EP remains similar to those of ES.

Particle Swarm Optimization

Introduction

The particle swarm optimization (PSO), in comparison the evolutionary programming and evolution strategies, is a newer method. First version of particle swarm optimization was presented in 1995 by James Kennedy and Russell Eberhart [23]. It was discovered through simulation of a simplified social model, and has roots in two main methodologies – it closely ties to artificial life: bird flocking, fish schooling and swarming theory, but even though it has emerged from different field, relates to evolutionary algorithms.

Bird flocking was an interesting field of research and simulation. Most models of flocking behavior relied on manipulation of inter-individual distances – the synchrony of flocking behavior was seen as a function of birds' efforts to maintain an optimum distance between themselves and their neighbors. On the other hand, sociological investigation of swarm behavior suggested that individual members of the swarm can profit from discoveries and experience of other swarm members, or in other words – social sharing of information provides an evolutionary advantage. This was the fundamental hypothesis for development of particle swarm optimization, motivated by a wish for developing a model of human social behavior. Human behavior is, of course, far more complicated than bird flocking or fish schooling – besides moving in three-dimensional space, humans change in multidimensional abstract space, collision free.

The conceptual development of PSO began as a simulation of simplified social milieu. The original idea was to simulate choreography of bird flock in search for food – the simulated birds were trying to find the best feeding position in two-dimensional space. Each position in the search space has a fitness function.

First methods were using nearest-neighbor velocity matching. Later the "cornfield" was introduced and the space was valued by means of a fitness function – each point in space has a value that measures the "goodness" of the point, in this case – amount of food.

Moreover, i-th bird in the group knew the globally best feeding position found by some member of the group $\mathbf{p}_g$, but also remembered the best feeding position that it has found itself $\mathbf{p}_i$. The difference between current position and these positions was multiplied by a random factor and added to current velocity of the bird. This has changed the visual appearance of the flock in the simulation, and finally the algorithm was named particle swarm. Both $\mathbf{p}_i$ and $\mathbf{p}_g$ have very significant role in explanation of PSO model. Conceptually, $\mathbf{p}_i$ resembles autobiographical memory, as each individual remembers its own experience – and it tends to return to the place that most satisfied it in the past. On the other hand $\mathbf{p}_g$ is similar to publicized knowledge, a group norm or standard which individuals seek to attain. Incremental influence of $\mathbf{p}_i$ and $\mathbf{p}_g$ is also significant – dominant influence of $\mathbf{p}_i$ (private best) results in excessive wandering through search space, whereas dominant influence of $\mathbf{p}_g$ can trap the swarm in the local optima. A delicate balance between conservative testing

of known regions and risky exploration of the unknown is required. The experiments were conducted on a model of a flock and later changed to multidimensional form.
The PSO outline is following: firstly, the positions and velocities of each particle in the swarm are chosen randomly. In each step, $\mathbf{p}_i$ and $\mathbf{p}_g$ are updated. Afterwards, velocities of particles are updated. The initial version of updates velocities of the particles in each algorithm step according to this formula:

$$\mathbf{v}_{i,t+1} = \mathbf{v}_{i,t} + 2 \cdot \varphi_1 \cdot (\mathbf{p}_i - \mathbf{p}) + 2 \cdot \varphi_2 \cdot (\mathbf{p}_g - \mathbf{p})$$

where φ is a random number sampled from uniform distribution and $\mathbf{p}$ current position of the particle. This in essence propels the individual to weighted average of $\mathbf{p}_i$ and $\mathbf{p}_g$.

The particle swarm can be viewed as a mid-level form of artificial life, occupying the space between evolutionary search which takes eons and neural processing that happens in order of milliseconds. The usage of fitness concept and relying on stochastic processes strongly relates it to evolutionary computation, but the uniqueness of PSO was 'flying' of potential solutions through hyperspace and accelerating towards better solutions, while other evolutionary computation methods operate directly on potential solutions represented as locations in hyperspace.

Particle trajectories, constriction, space of states

Particle swarm optimization implementations have proven that the algorithm performs well – empirical evidence accumulated that the algorithm is very useful tool for optimization, but it wasn't adequately explained how it works until much later. Insights in algorithm behavior [24] indicated that the algorithm is susceptible to *explosion* – movement of particles towards infinity. The attempts to avoid this behavior were mostly related to limiting the maximum particle speed by setting the upper limit for $\|\mathbf{v}\|$.

Maurice Clerc and James Kennedy's article from 2002 [25] presents a significant theoretical breakthrough in analysis of PSO giving the generalized model along with methods for controlling convergence properties. The analysis starts with highly simplified version of the particle swarm. The algorithm is approached from the point of view of a particle.
The analysis begins with deterministic version of algorithm – the swarm is reduced to single particle, the problem is made one-dimensional and the equations are further simplified by making the random number and the so-far-found best position constant. Using these simplifications, the system of equations is:

$$\begin{cases} v_{t+1} = v_t + \varphi \cdot (p - x_t) \\ x_{t+1} = x_t + v_t \end{cases}$$

where p and φ are constant. Substituting $y = p - x$ gives

$$\begin{cases} v_{t+1} = v_t + \varphi \cdot y_t \\ x_{t+1} = -v_t + (1 - \varphi) \cdot y_t \end{cases}$$

The analysis starts with algebraic point of view – discrete time domain. The matrix of given system is

$$M = \begin{bmatrix} 1 & \varphi \\ -1 & 1-\varphi \end{bmatrix}$$

and the eigenvalues of the system are

$$\begin{cases} e_1 = 1 - \frac{\varphi}{2} + \frac{\sqrt{\varphi^2 - 4\varphi}}{2} \\ e_2 = 1 - \frac{\varphi}{2} - \frac{\sqrt{\varphi^2 - 4\varphi}}{2} \end{cases}$$

Details can be found in the original article [25]. Obviously, the value of φ=4 is important. Depending on the chosen value of φ trajectory of the particle can be cyclic or quasi-cyclic for φ<4, strictly monotonously increasing if that is not satisfied, with somewhat deceptive behavior for φ=4 in first iterations. These conclusions derive from highly simplified model and discrete time. Switching to continuous time domain and the analytic point of view gives the second-order differential equation:

$$\frac{\partial^2 v}{\partial t^2} + \ln(e_1 e_2)\frac{\partial v}{\partial t} + \ln(e_1)\ln(e_2)v = 0$$

where e_1 and e_2 are the roots of $\lambda^2 + (\varphi - 2)\lambda + 1 = 0$:

$$\begin{cases} e_1 = 1 - \frac{\varphi}{2} + \frac{\sqrt{\varphi^2 - 4\varphi}}{2} \\ e_2 = 1 - \frac{\varphi}{2} - \frac{\sqrt{\varphi^2 - 4\varphi}}{2} \end{cases}$$

General solution for v is then $v(t) = c_1 e_1^t + c_2 e_2^t$ and similar expression is produced for $y(t)$

$y(t) = \frac{1}{\varphi}\left(c_1 e_1^t (e_1 - 1) + c_2 e_2^t (e_2 - 1)\right)$ where coefficients *c1* and *c2* depend on initial values of v and y (for t=0).

When e_1 is not equal to e_2 ($\varphi \neq 4$), we have:

$$\begin{cases} c_1 = \frac{-\varphi y(0) - (1 - e_2)v(0)}{e_2 - e_1} \\ c_2 = \frac{\varphi y(0) + (1 - e_1)v(0)}{e_2 - e_1} \end{cases}$$

For $\varphi = 4$

$$\begin{cases} v(0) = c_1 + c_2 \\ y(0) = -\frac{c_1 + c_2}{2} \end{cases}$$ so $v(0) + 2y(0) = 0$ must be satisfied to prevent discontinuity.

Considering expressions e_1 and e_2 and eigenvalues of the matrix M given above, the same discussion about the sign of $\left(\varphi^2 - 4\varphi\right)$ can be made about the existence of cycles. This time analytic point of view with continuous time provides the guideline for *preventing explosion* of the particle swarm: $\max\left(\left|e_1\right|,\left|e_2\right|\right) > 1$

More general implicit representation (IR) is produced by adding five *coefficients* $\{\alpha,\beta,\gamma,\delta,\eta\}$ that define the system in five-dimensional space. Using these coefficients, the system becomes

$$\begin{cases} v_{t+1} = \alpha v_t + \beta\varphi y_t \\ y_{t+1} = -\gamma v_t + (\delta - \eta\varphi) y_t \end{cases}$$

$$\varphi \in R_+^*$$

$$\forall t \in N, \{y_t, v_t\} \in R^2$$

The matrix of the system is

$$M' = \begin{bmatrix} \alpha & \beta\varphi \\ -\gamma & \delta - \eta\varphi \end{bmatrix}$$ where e_1' and e_2' are its eigenvalues.

The explicit (analytic) representation is

$$\begin{cases} v(t) = c_1 \left(e_1'\right)^t + c_2 \left(e_2'\right)^t \\ y(t) = \dfrac{1}{\beta\varphi}\left(c_1 \left(e_1'\right)^t (e_1' - \alpha) + c_2 \left(e_2'\right)^t (e_2' - \alpha)\right) \end{cases}$$

$$\varphi \in R_+^* \qquad \forall t \in N, \{y(t), v(t)\} \in R^2$$

The coefficients are dependent on initial values of *y* and *v*.

$$\begin{cases} c_1 = \dfrac{-\beta\varphi y(0) - (\alpha - e_2')v(0)}{e_2' - e_1'} \\ c_2 = \dfrac{\beta\varphi y(0) + (\alpha - e_1')v(0)}{e_2' - e_1'} \end{cases}$$

The *constriction coefficients* χ_1 and χ_2 are defined by

$$\begin{cases} e_1' = \chi_1 e_1 \\ e_2' = \chi_2 e_2 \end{cases}$$ where $$\begin{cases} e_1 = 1 - \dfrac{\varphi}{2} + \dfrac{\sqrt{\varphi^2 - 4\varphi}}{2} \\ e_2 = 1 - \dfrac{\varphi}{2} - \dfrac{\sqrt{\varphi^2 - 4\varphi}}{2} \end{cases}$$ are the eigenvalues of the original system.

The implicit representation considers *t* as an integer and *v*(*t*) and *y*(*t*) are real numbers, whereas for explicit representation these are real if and only if *t* is an integer. However – the above equations are defined for any $t \in R^+$, so *v*(*t*) and *y*(*t*) can be true complex numbers.

This enables visualizing the algorithm performance in five-dimensional space of states. Further details and relations linking the values of five added coefficients are given in the original article.

There are several particular classes of system coefficients that are particularly interesting. Class 1 model, $\begin{cases}\alpha=\delta\\ \beta\gamma=\eta^2\end{cases}$, class 1' model $\begin{cases}\alpha=\beta\\ \gamma=\delta=\eta\end{cases}$, class 1'' $\alpha=\beta=\gamma=\eta$ and class 2 model $\begin{cases}\alpha=\beta=2\delta\\ \eta=2\gamma\end{cases}$.

Depending on the chosen parameter set, the system might have a discontinuity in φ due to the presence of square root in the eigenvalues. To have a completely continuous system, the discriminant must be positive. By taking into account that the values must be positive to be plausible, this renders the following condition

$$\beta\gamma<\eta(\alpha-\delta)$$

The general criterion for convergence becomes

$$\begin{cases}\left|e_1^{'}\right|<1\\ \left|e_2^{'}\right|<1\end{cases}$$

$v(t)$ and $y(t)$ are true complex numbers and the whole system can be represented in a 5-dimension space $(\mathrm{Re}(y),\mathrm{Im}(y),\mathrm{Re}(v),\mathrm{Im}(v),\varphi)$.

By investigating the system without complex values, analyzing only $(\mathrm{Re}(y),\mathrm{Re}(v),\varphi)$, it is obvious that system exhibits spiral convergence for φ<4, difficult convergence for φ approximately 4 (there is a discontinuity for φ=4), and quick, almost linear convergence for φ=4.

When observed in the complex space, the oscillating behavior seen just by looking at the real part of velocities and positions actually becomes continuous spiral movement. The attractor is dependent on the constriction coefficients.

All these conclusions were taken on grounds of constant φ. After generalization –returning to random values of φ, the PSO with one constriction coefficient has the following form:

$$\begin{cases}\mathbf{v}(t+1)=\chi\left(\mathbf{v}(t)+\varphi_1(\mathbf{p}_i-\mathbf{x}(t))+\varphi_2(\mathbf{p}_g-\mathbf{x}(t))\right)\\ \mathbf{x}(t+1)=\mathbf{v}(t+1)+\mathbf{x}(t)\end{cases}$$

The constriction must be chosen carefully – even though the appropriate choice of constriction coefficients ensures convergence, i.e. that the velocities of the particles will end at being equal to 0, but it does not guarantee that the convergence point is the optimum. Therefore, constricting the system too strong means reducing its “oscillation” completely and even if it prevents explosion, it would obviously also prevent *exploration* of the search space. In other words, *moderately* constricting the system by allowing the particles to explore the search space should be the chosen approach.

The particle swarm using constriction coefficients in principle does not differ from the particle swarm without them – the coefficients $\{\alpha, \beta, \gamma, \delta, \eta\}$ are calculated prior the algorithm start. The main benefit of the aforementioned theoretical research is that the particle system's explosion can be controlled *without* resorting to any arbitrary or problem specific parameters. In other words, explosion is controlled at the algorithmic level and convergence of the algorithm is assured.
Usage of constriction does not excessively modify the original algorithm – the coefficients are calculated upon the algorithm start from the chosen constriction type and the eigenvalues of the particle system matrix. Constriction can assure convergence, whereas
To be convergent, the constricted system makes limitation of the velocities unnecessary. A liberal limit of velocity, though, helps convergence – as proven by Eberhart and Shi [26]. The liberal limit means, for instance, that the particle with maximum speed can travel across the whole search space in a single iteration. Constriction coefficients have been developed from the individual particle's point of view and regulating the constriction means regulating dynamic characteristics of the particle swarm, tuning exploration versus exploitation. Moreover, some earlier theoretical observations have quite detailed explanation in the 5-dimensional space of states.

Informed swarms and communication schemes

PSO reportedly works well due to its social dimension – the real strength of the swarm is derived from the interaction among particles as they search the space collaboratively. When the collaborative effect is removed from the algorithm, i.e. the particles do not know anything about the global best position; the algorithm performance is heavily degraded [27]. In these versions of the PSO there were no communication connections between particles. Since the communication was neglected, the model was called *cognitive* model: the particles relied only on the information they gathered alone.
Recent research in the field of particle swarm optimization was directed towards investigating viable communication schemes. Claiming that premature convergence has to do with too fast propagation of information and that a lot of information gathered by *other* particles, not only the one that has currently found the best particle, Rui Mendes and James Kennedy [28] have introduced a *fully informed* particle swarm (FIPS). The canonical version of the particle swarm algorithm works by searching iteratively in a region defined by each particle's best previous success, the best previous success among the neighboring particles and the previous velocity. Their work starts from the Clerc's constricted particle swarm.

$$\begin{cases} \mathbf{v}(t+1) = \chi\left(\mathbf{v}(t) + \varphi_1(\mathbf{p}_i - \mathbf{x}(t)) + \varphi_2(\mathbf{p}_g - \mathbf{x}(t))\right) \\ \mathbf{x}(t+1) = \mathbf{v}(t+1) + \mathbf{x}(t) \end{cases}$$

The variation in the "classic" particle swarm introduced in several ways – by random numbers φ, the difference between the current and the previous best position, the updating of the global best position. Actually, the most important source of variation in a traditional particle swarm is the difference between particle's previous best $\mathbf{p}_i$ and global best $\mathbf{p}_g$, while

random weighting only keeps the particle searching between and beyond region defined by these two points.
Traditional particle swarm chooses one neighbor to be the source of influence and ignores the other. On the contrary, fully informed particle swarm takes into account best positions and distance to all the particles in neighborhood – hence the name fully informed particle swarm. All the members of the neighborhood contribute to velocity adjustment.
The neighborhood size and sociometry of the neighborhood hence becomes more important – it determines how diverse the influences will be. The topological structure of the population controls the exploration/exploitation behavior of the swarm. In other words, since the behavior of each particle is affected by local neighborhood, the topology affects the search at the low level by defining neighborhoods.
The topologies can be represented as graphs, its nodes are particles and vertices represent links between the particles able to exchange information directly. Several topologies are of particular interest:

- *all* or *star* topology, developed in the first days of PSO, when every particle is directly connected to any other particle, the graph is fully connected so the information about the best position spreads very quickly
- *ring* topology, the common alternative to the previous, where the graph has a minimum number of edges, the information travels slowly since each particle is linked only to its direct neighbors
- *four clusters* topology, where the particles are divided in 4 cliques connected by several gateways, representing isolated communities where few individuals have acquaintance outside the group
- *pyramid* – three dimensional wireframe pyramid, it has the lowest average distance and the highest first and second degree neighbors.

The experiments show that introducing more information improves the performance classic particle swarm – however, there is still both theoretical and empirical research to be done. Generally, increasing the neighborhood size seems to deteriorate the performance of the swarm, so the *all* topologies perform the worst, compared to others. Interestingly, including the particle's own best sometimes seems to have a negative effect – the informed swarm that excludes particle's own best information performs better. This might indicate that the information already contained in the vector of speed doesn't need to be further emphasized by including the particle's best location. Another promising characteristic of the FIPS is that it is relatively independent of the initial initialization – the asymmetrical initialization of particle swarm to the lesser extent degrades the fully informed swarm performance.
FIPS emphasizes [29] that the best-of-neighborhood location information is not enough – the information conveyed by distances between particles is also participating and the particle is not being drawn towards the best-of-neighborhood, instead it is being adjusted by a kind of average difference between *each* neighbor's previous best and the target particle's current position. However, recent work shows that this approach might have degrading effect in some fitness landscapes so that the "classic" methods outperform the FIPS. The

EPSO – Evolutionary Particle Swarm Optimization

Introduction

The shortest definition of evolutionary particle swarm algorithms is: EPSO algorithms are evolutionary methods that borrow the movement rule from PSO and use it as a recombination operator that evolves under pressure of selection. By using this hybrid approach, an algorithm is built that has already empirically proven its superiority over classical approaches, both in evolutionary algorithms and PSO.
Hybrid algorithms are common in the soft-computing – it is often hoped that the hybrid will combine the good traits of both methods and therefore build a more powerful method. It might seem easy and convenient to describe EPSO simply as a variant of particle swarm optimization, using the vocabulary and concepts of PSO to describe EPSO. However, EPSO not only has ancestry in PSO – there are mechanisms in EPSO that are clearly related to those in evolutionary algorithms, and keeping that perspective on EPSO should lead to better understanding on how and why it does work.
The basic outline of EPSO is:

- replication - each particle is replicated r times
- mutation - each particle has its strategic parameters mutated
- reproduction - each mutated particle generates an offspring through recombination, according to the particle movement rule, described below
- evaluation - each offspring has its fitness evaluated
- selection - by stochastic tournament or other selection procedure, the best particles survive to form a new generation, composed of a selected descendant from every individual in the previous generation

Recombination and movement rule

Given a particle in generation *k*, a new particle in generation *k+1* is reproduced using the following rule, similar to PSO rule:

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} + \mathbf{v}_i^{(k+1)}$$
$$\mathbf{v}_i^{(k+1)} = w_{i1}^* \mathbf{v}_i^{(k)} + w_{i2}^* (\mathbf{p}_i - \mathbf{b}_i) + w_{i3}^* (\mathbf{p}_g^* - \mathbf{x}_i)\mathbf{P}$$

where the values are

$\mathbf{p}_i$ – best point found by particle i in its past life up to the current generation

$\mathbf{p}_g$ – best overall point found by the swarm of particles in their past life up to the current generation

$\mathbf{x}_i^{(k)}$ – location of particle *i* at generation k

$\mathbf{v}_i^{(k)} = \mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k-1)}$ – is the velocity of particle *i* at generation k

w_{i1} – weight conditioning the *inertia* term (the particle tends to move in the same direction as the previous movement)

w_{i2} – weight conditioning the *memory* term (the particle is attracted to its previous best position)

w_{i3} – weight conditioning the *cooperation* or *information exchange* term (the particle is attracted to the overall best-so-far found by the swarm).

P – communication factor, a diagonal matrix affecting all dimensions of an individual, containing binary variables of value 1 with probability p and value 0 with probability (1-p); the p value, set as an external parameter, controls the passage of information within the swarm and is 1 in classical formulations. This reproduction is illustrated in the following figure.

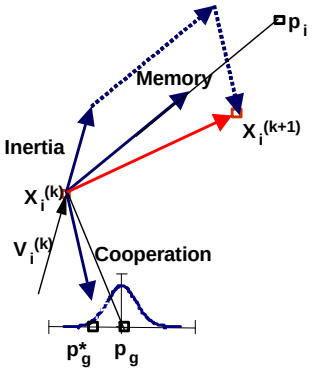

Figure 1 – Reproduction rule of EPSO

The symbol * in the formulation indicates that these parameters will undergo *evolution* under a mutation. The recombination rule is called movement rule in PSO, but it is a form of intermediary recombination operator.

If using the perspective of evolutionary algorithm, the recombination operator is specific due to the choice of parents: the global best, the best particle ancestor and the direct parent. This means that, for practical purposes, this method includes a provision for elitism, because the particle best ancestor and the global best are kept from generation to generation. Moreover, in this method, the recombination operator is adaptive and evolving, instead of being fixed.

Mutation of strategic parameters

The basic rule for mutating the strategic parameters is inspired by similar rule in evolutionary algorithms, where it has proven its efficiency:

$$w_{ik}^{*} = w_{ik}\ \left[\log \mathrm{N}(0,1)\right]^{\tau}$$

The weights are mutated by means of lognormal mutation conditioned with learning parameter τ.

Global best solution is also disturbed randomly:

$$\mathbf{p}_g^{*} = \mathbf{p}_g + w_{i4}^{*}\mathbf{N}(0,1)$$

where w_{i4} is the fourth strategic parameter associated with each particle (or individual, in vocabulary of evolutionary algorithms). This strategic parameters controls the disturbance of *up-to-now* global best, so that its neigborhood could be explored, assuming that the real global optimum is not yet found during the process.

Control of communication among particles

As it was mentioned above, in the research of PSO the topology plays a significant role for the algorithm efficiency. In EPSO, stochastic scheme oscillates between purely cognitive model and the star model where every particle is aware of global optimum. This is not adaptive scheme of communication, but an alternative way of taking advantage of *slower* propagation of information between particles. Further research will be done in this direction.

Constrained selection and inherent elitism

In all versions of the EPSO family so far, the selection operator does not act freely on the whole population. The replication phase generates *r* clones of a particle (including in this count the original one); these undergo mutation in their strategic parameters and then the recombination operator (movement rule) generates *r* offspring in *r* different locations.

The selection operator acts on *these* descendants and chooses *one* survivor for the following generation. This procedure is repeated for all individuals.

In a selfish or cognitive model, with no communication among particles, this would result in n independent evolutionary processes, but the communication term of the recombination operator avoids this. Therefore, it is not a parallel evolutionary method – but it could be seen as a parallel method for particle swarms. One also observes that elitism is present in the model, because the information about the best ancestors is kept and used in recombination to form new particles. When an individual does not directly generate descendants but contributes to the formation of new individuals via recombination, we can call it *dormant*. The following figure illustrates the process of selection.

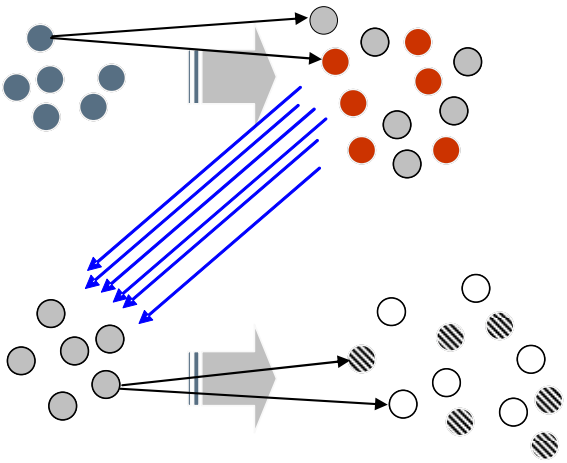

Figure 2 - Illustration of reproduction/ selection process for *r*=2

Benefits of EPSO

The convergence of both ES and EP has been theoretically investigated. The generation of offspring is governed with mutation and recombination, but the *push* towards the optimum is the role of selection operator. Classical PSO schemes, on the other hand, rely on the reproduction scheme whereas selection is trivial – one parent generates one descendant. It is the reproduction rule – or, in vocabulary of PSO, movement rule that assures the progress towards the optimum.

EPSO exploits both of these mechanisms in sequence, each one with its own probability of producing better individuals and also better group on average. If EPSO is seen as a evolutionary algorithm, its recombination is already biased to push the algorithm towards the optimum. Selection takes place afterwards, acting on generation that's already better than the parent one and having the additive effect.

Self-adaptation is another important characteristic of EPSO – it avoids in large scale the need for fine-tuning the parameters of the algorithm. These characteristics in conjunction give robustness to EPSO models and make it applicable to real-life problems [31].

Directions of theoretical research for EPSO

Evolutionary algorithms have proved to be powerful meta-heuristic solvers when applied to complex problems and self-adaptive models have proved to improve the efficiency of the algorithms in most cases. In the evolutionary algorithms, self-adaptive characteristics are usually given to them by manipulating (under selection) mutation rates. The EPSO algorithms give self-adaptive characteristics by manipulating (under selection) the recombination operator. Therefore, when looking from perspective of EA, EPSO represents an evolutionary algorithm with the recombination rule from particle swarm optimization, reinterpreted as a form of intermediate recombination. The pressure towards the optimum is given not only by selection mechanism but also by the properties of the reproduction

mechanism. EPSO can also be viewed as a specific case of particle swarm optimization algorithm with evolving weights and stochastic disturbance of communication schemes.
Considering the theoretical research of EPSO, it can be done from two points: either by applying the methods and measures from the world of evolutionary algorithms (for instance, the progress rate), or by extending the work in researching the PSO.
Both of these research directions provide incentives for research related to EPSO – however, the evolutionary point of view seems more promising. EPSO has several characteristics that could hardly be described from the world of PSO, most notably the creation of multiple offspring. Measures and analyses, similar to progress rate analyses in the world of EA, should be the right direction for investigating the world of EPSO. The PSO point of view, isn't unusable – it provides promising fields for improvement regarding sociometry and communication schemes between particles. These could be exploited in future versions of EPSO, so an important direction of research also relates to communication schemes in EPSO and possibility of evolving these schemes, finding thus the optimal way of propagating the information through the swarm.

References

[1] Miranda, V., "Evolutionary Algorithms with Particle Swarm Movements," *Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on* , pp. 6- 21, 6-10 Nov. 2005

[2] Wolpert, D.H.; Macready, W.G., "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on* , vol.1, no.1, pp.67-82, Apr 1997

[3] Bäck, T., Fogel; D.B., Michalewicz, T. (editors), "Evolutionary Computation 1: Basic Algorithms and Operators", Institute of Physics Publishing, 2000

[4] Schwefel, H.P.; Rudolph, G., "Contemporary Evolution Strategies", *3rd International Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, 1995.

[5] Beyer, H.G., "Towards a Theory of Evolution Strategies: Some Asymptotical Results from the (1, λ) Theory", *Evolutionary Computation, vol.1*, 1993

[6] Beyer, H.G., "Towards a Theory of Evolution Strategies: the (μ,λ) Theory", *Evolutionary Computation*, vol.1, 1993.

[7] Beyer, H.G., "Towards a Theory of Evolution Strategies: Progress Rates and Quality Gain for (1, λ) strategies on (Nearly) Arbitrary Fitness Functions", *Parallel Problem Solving from Nature*, Springer-Verlag, 1994.

[8] Schwefel, H.P., "Natural Evolution and Collective Optimum Seeking", in *Computational System Analysis:Topics and Trends*, Elsevier, Amsterdam, 1992.

[9] Beyer, H.G., The Theory of Evolution Strategies, Natural Computing Series, Springer, 2001.

[10] Sendhoff, B.; Kreuz, M.; von Seelen, W., "A Condition for the genotype-phenotype mapping: Causality", *Proceedings of 7th International Conference on Genetic Algorithms*, Morgan-Kaufmann, 1997.

[11] Goldberg, D., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley, 1989.

[12] Beyer, H.G., "Towards a Theory of Evolution Strategies, on benefit of sex, the (μ/μ, λ)-theory", *Evolutionary computation* vol. 3, 81-111, 1995.

[13] Beyer, H.G., Schwefel, H.P., "Evolution Strategies: A Comprehensive Introduction", *Natural Computing 1*, pp.3-52, Kluwer Academic Publishers, 2002.

[14] Beyer, H.G., "Towards a Theory of Evolution Strategies: Self-Adaptation", Evolutionary *Computation*, vol. 3, no. 3, pp. 311-347, 1996.

[15] Beyer, H.-G.; Deb, K., "On self-adaptive features in real-parameter evolutionary algorithms ," *Evolutionary Computation, IEEE Transactions on*, vol.5, no.3, pp.250-270, Jun 2001

[16] Meyer-Nieberg, S.; Beyer, H.-G., "On the analysis of self-adaptive recombination strategies: first results," *Evolutionary Computation, 2005. The 2005 IEEE Congress on* , vol.3, pp. 2341- 2348 Vol. 3, 2-5 Sept. 2005

[17] Gruenz, L.; Beyer, H.-G., "Some observations on the interaction of recombination and self-adaptation in evolution strategies," *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* , vol.1, no.pp.-645 Vol. 1, 1999

[18] Ostermeier, A.; Gawelczyk, A.; Hansen, N., Step-size adaptation based on non-local use of selection information, *Parallel Problem Solving from Nature*, 3, p. 189–198. Springer-Verlag, 1994.

[19] Fogel, D.B., "Phenotypes, genotypes, and operators in evolutionary computation", *Evolutionary Computation, 1995., IEEE International Conference on*, vol.1, pp.193-, 1996.

[20] Fogel, D.B., "An introduction to simulated evolutionary optimization", *Neural Networks, IEEE Transactions on* , vol.5, no.1, pp.3-14, Jan 1994

[21] Fogel, D.B.; Fogel, L.J.; Atmar, J.W., "Meta-evolutionary programming," *Signals, Systems and Computers, 1991. 1991 Conference Record of the Twenty-Fifth Asilomar Conference on* , vol., no.pp.540-545 vol.1, 4-6 Nov 1991

[22] Fogel, David B.; Fogel, Gary B.M; Ohkura, K, "Multiple-vector self-adaptation in evolutionary algorithms", Biosystems, Volume 61, Issues 2-3, 8 July 2001, pp. 155-162.

[23] Kennedy, J.; Eberhart, R., "Particle swarm optimization," *Neural Networks, 1995. Proceedings., IEEE International Conference on* , vol.4, pp.1942-1948, 1995.

[24] Kennedy, J., "The particle swarm: social adaptation of knowledge," *Evolutionary Computation, 1997., IEEE International Conference on* , pp.303-308, 1997

[25] Clerc, M.; Kennedy, J., "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *Evolutionary Computation, IEEE Transactions on* , vol.6, no.1, pp.58-73, Feb 2002

[26] Eberhart, R. C.; Shi, Y., "Comparing inertia weights and constriction factors in particle swarm optimization", in *Proceedings of 2000 Congress of Evolutionary Computation*, San Diego, July 2000

[27] Kennedy, J., "The particle swarm: social adaptation of knowledge", *Evolutionary Computation*, *IEEE International Conference*, pp. 303-308, 1997

[28] Mendes, R.; Kennedy, J.; Neves, J., "The fully informed particle swarm: simpler, maybe better," *Evolutionary Computation, IEEE Transactions on* , vol.8, no.3, pp. 204-210, 2004

[29] Kennedy, J.; Mendes, R., "Neighborhood topologies in fully informed and best-of-neighborhood particle swarms," *Systems, Man and Cybernetics, Part C, IEEE Transactions on* , vol.36, no.4pp. 515- 519, July 2006

[30] Kennedy, J., "In Search of the Essential Particle Swarm," *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on* , vol., no.pp. 1694- 1701, 16-21 July 2006.

[31] Miranda V.; Oo N. W.; "New experiments with EPSO – Evolutionary Particle Swarm Optimization", *IEEE Swarm Intelligence Symposium 2006*, Indianapolis, Indiana, USA, 2006.