Stochastic Star Communication Topology in Evolutionary Particle Swarms (EPSO)

Vladimiro Miranda[1,2], Hrvoje Keko[1] and Álvaro Jaramillo Duque[1]
[1]INESC Porto, Institute of Engineering in Systems and Computers of Porto, Portugal
[2]FEUP, Faculty of Engineering of the University of Porto, Portugal

Abstract

This paper reports the results of the adoption of a probabilistically defined communication structure in a special algorithm coined as EPSO – Evolutionary Particle Swarm Optimization, which is classified as an evolutionary algorithm using a particle movement rule as the recombination operator. Alternatively, EPSO may be seen as an algorithm of the family of PSO (Particle Swarm Optimization) but with a self-adaptive mechanism applied to make the weights of the movement rule evolve improving the performance of the algorithm. The paper presents results showing that a probabilistically controlled communication (to the particles of a swarm) of the location of the *best-so-far* point leads to better convergence and that the optimal value of the probability of communication depends on the topology of the surface being searched. Also, full communication (similar to classical PSO) has in all cases been shown to be worse than probabilistically constrained communication. This is demonstrated by comparing results in different test functions and also in the application of EPSO to an industrially relevant application – the reactive power planning in large scale power systems.

Introduction

There are two mechanisms necessary for finding the optimum solution of a given problem in the context of a search algorithm: a *movement* generator (to produce new candidate solutions) and an *evaluation* procedure of alternatives in the search space. The simplest and perhaps least efficient search method is a purely random search, where the evaluation of sampled points does not influence the selection of new points.

Meta-heuristic methods employ smart strategies for searching in the solution space. For example, in evolutionary algorithms (EA) the movement mechanism is constituted by the action of mutation+recombination over alternatives, because the concurrence of these two operators proposes new points in space departing from previous locations, which are then subject to evaluation and selection. It is the presence of a *selection* operator (assuming competition among alternatives to identify which will become the departing points or seeds to generate new points) that distinguishes evolutionary algorithms from other meta-heuristics.

Particle swarm optimization (PSO) algorithms have no selection operator but adopt a specific movement rule that defines how a new particle is created departing from its history and from information from the swarm. This by itself, under controlled circumstances, drives the swarm to the optimum of a problem. The PSO movement rule has dynamic characteristics that drive the optimization process towards the optimum and does not require selection (one could say that selection is trivial). Eberhart and Kennedy developed PSO proposing it as an analogy to swarms of birds or schools of fish [1] – in PSO, individuals may exchange information influencing the outcome of the movement mechanism. This is a concept far from the classical paradigm of EA.

We have said that, in evolutionary algorithms, the movement mechanism is composed of two operators: mutation and recombination. Mutation is a unary operator, acting on a single individual and responsible for generating a new solution – a new individual – by applying random modifications to it. Recombination generates new individuals (new points in the search space) by randomly mixing characteristics from more than one parent (solutions previously found). Some EA give more relevance to the mutation operator while others rely mostly on the recombination operator. These operators, classically defined, are neutral and do not contribute to pushing the search towards optimum: this task is left to the selection rule.

PSO has no mutation operator. However, it is possible to understand the PSO movement rule as a specific form of chromosome recombination. This interpretation derives from the evolutionary concept that is characteristic of EPSO.

This movement/recombination rule has, on its own, remarkable property of pushing the population towards the optimum, as PSO algorithms have demonstrated. This effect may be added to the action of a selection operator to obtain a cumulative result improving the performance of the meta-heuristic as compared to alternatives that rely on a single push either from the movement rule or from the selection operator.

Evolutionary Algorithms having as recombination rule the movement rule of PSO have been called EPSO or Evolutionary Particle Swarm Optimization algorithms [2]. In particular, the basic version of EPSO is a self-adaptive algorithm or, more clearly said, an algorithm that has a self-adaptive recombination mechanism. EPSO models have been favorably benchmarked against other evolutionary algorithms and especially against PSO models, in laboratory and in real world problems, with applications in Power Systems. Recent results may be found in [3] where comparisons with the performance of the latest to date standard version of PSO (called Standard PSO 2006 [4]) consistently show a superior performance of EPSO.

In the following sections, the paper presents recent developments in EPSO and discusses the improvement achieved by adopting a stochastic star communication topology, first suggested in [5][6], instead of the deterministic scheme usually adopted in PSO. By *stochastic star* we mean that the knowledge of each particle of the whereabouts of the global best-so-far point is randomly controlled by a communication probability p, instead of being deterministic as in the classical PSO formulations (like having p = 1) using the star communication scheme. The effect is that a particle will ignore the global best in some iterations (as in the selfish or cognitive model) and take it in account in other iterations. This not only allows the inertia+memory factors to singly determine the trajectory of the particle in a sequence of iterations before receiving another pull from the global best, but also delays the impact of updated information on the location of the global best, achieving an effect somewhat similar to the one present in ring communication schemes.

In many of our experimental results, a communication probability of p between 0.1 and 0.4 led to better results than a classical deterministic star model equivalent to having p = 1. One is led to the conjecture that restraining the free flow of information about the global best allows more local search by each particle, eliminates disturbing noise, allows the dynamics of particle movement to be more stable and avoids premature convergence. As it is easily observed, this is yet another way of acting on the recombination operator.

This paper presents evidence of the improvement achieved with the stochastic star scheme both in the case of test problems and in the case of application to power system problem of reactive power planning. It is important, in order for the reader to easily grasp the concepts in this paper, to allow flexibility in jumping back and forth from particle swarm language and concepts to evolutionary algorithm vocabulary and concepts. PSO conceptually depicts a particle as an object traversing space, while EA thinks of individuals (parents and offspring) as independent objects. However, PSO is method discrete in time and the so called trajectory of a particle is, in fact, a sequence of points, locations or alternatives in space, and each one may be considered as the offspring of the previous one – the EA interpretation would then see the trajectory as a sequence of individuals in successive generations. Conversely, in an EA environment, if one follows an individual back in time from ancestor to ancestor, one may also draw a "generational" path across space that would be the search pattern for a given object.

EPSO viewed as an Evolutionary Algorithm

A general Evolutionary Algorithm may be organized in the following steps:

Procedure EA
initialize a random population P of μ elements
REPEAT
reproduction (introduce stochastic perturbations in the new population) – generate λ offspring…
…by *recombination*
…by *mutation*
evaluation - calculate the fitness of the individuals
selection - of μ survivors for the next generation, based on the fitness value
test for termination criterion (based on fitness, on number of generations, etc.)
Until test is positive
End EA

This scheme applies to all variants of EA, whether Genetic Algorithms, Evolutionary Programming or Evolution Strategies or other of the kind. As noted before, the driving force behind any EA is the selection operator. For the algorithm to perform a search, it requires the action of the replication or reproduction operators – recombination and mutation – to generate new points in space to be evaluated. The definition of the mutation operator is dependent of the particular problem under analysis.

In the implementations where an individual consists of a string of real values, the mutation procedure causes random perturbations on each variable. The mutation amplitude is governed by the variance of the distribution regulating mutations. When Gaussian mutations are considered, the standard deviation is often called *learning rate*. Self-adaptive Evolutionary Algorithms have the learning rates adopted as strategic variables and added to the chromosome together with the object variables [7][8]. During the process, the strategic parameters are themselves subject to mutation and selection and eventually they acquire values that allow a near optimal progression rate towards the optimum [9]. It should be noted that self-adaptation has been used mainly on the mutation operator even if some attempts have been made to create adaptive crossover scheme – a good review is found in [10].

Recombination generates a new individual by combining features from a set of individuals in the population. For the recombination many forms have been proposed, one of most commonly used being intermediary recombination where the value of any variable in the offspring receives a contribution from all parents. The child value of variable is a result from a method of averaging values of a certain number of parents. Various schemes for choosing parent individuals and adding their contribution exist, including ones that adopt weighted averages with random choosing of weights. The general scheme of the particle swarm movement rule can be examined from this standpoint.

The basic particle *swarm movement rule*, producing a new individual $\mathbf{X}_i$ for iteration (k+1) is based on

$$\mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + \mathbf{V}_i^{(k+1)}$$

where $\mathbf{V}_i$ is called the particle i velocity and is defined by

$$\mathbf{V}_i^{(k+1)} = \mathbf{A}\mathbf{V}_i^{(k)} + \mathbf{B}(\mathbf{b}_i - \mathbf{X}_i^{(k)}) + \mathbf{C}(\mathbf{b}_G - \mathbf{X}_i^{(k)})$$

where the first term of the summation represents inertia or habit, the second represents memory and the third represents cooperation or information exchange. The parameters **A, B, C** are diagonal matrices (of a dimension of the search space) with weights fixed in the beginning of the process, $\mathbf{b}_i$ is the particle i past best position and $\mathbf{b}_G$ is best location found by the swarm. In a classical particle swarm formulation, parameter **A** is affected by a decreasing value over time (iterations), while the parameters **B** and **C** are multiplied by random numbers sampled from a uniform distribution in [0,1].

A different aspect to the movement rule is given by modifying the summation:

$$\mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + \mathbf{A}(\mathbf{X}_i^{(k)} - \mathbf{X}_i^{(k-1)}) + \mathbf{B}(\mathbf{b}_i - \mathbf{X}_i^{(k)}) + \mathbf{C}(\mathbf{b}_G - \mathbf{X}_i^{(k)})$$

$$\mathbf{X}_i^{(k+1)} = (1 + \mathbf{A} - \mathbf{B} - \mathbf{C})\mathbf{X}_i^{(k)} - \mathbf{A}\mathbf{X}_i^{(k-1)} + \mathbf{B}\mathbf{b}_i + \mathbf{C}\mathbf{b}_G$$

This equation shows two things. First, there are four contributors to generate an offspring $\mathbf{X}_i^{(k+1)}$:

- its direct ancestor $\mathbf{X}_i^{(k)}$
- the ancestor of its ancestor $\mathbf{X}_i^{(k-1)}$
- a (possibly distant) past best ancestor $\mathbf{b}_i$
- the current global best known to this particle $\mathbf{b}_G$

Second, the sum of the parameters multiplying the four contributors (ancestors) is equal to 1. This expression can be viewed as a specific version of intermediary recombination in EA with $\mu = 4$ parents coupled with a special rule to determine who the parents are (in an intermediary recombination, each variable of the offspring receives a value that results from contributions from all the parents). The parents of a new particle $\mathbf{X}_i^{(k+1)}$ are being chosen from an *enlarged population* that includes not only the active particles but also the direct ancestors and the set of the past best ancestors.

This recombination rule has the property of pushing the population towards the optimum – in other words it may be biased towards the optimum and not neutral. EPSO algorithms put this rule together with a selection mechanism thus contributing to a double push towards the optimum.

To determine the best values to use in **A**, **B** and **C**, EPSO relies on a self-adaptive mechanism. **A**, **B** and **C** are

considered as strategic parameters, in the language of Self-Adaptive EA [9], and they are subject to selection. Under the pressure of selection they evolve to values adapted to the landscape being searched. This means that when an exploration mode is more profitable to a particle, some parameters are likely to be selected to exhibit large values while when an exploitation mode is more rewarding some parameters may fade away because smaller values give competitive advantage to the particle descendants.

Given a population as a set of particles, the general scheme of EPSO is the following:

- REPLICATION - each particle is replicated r times
- MUTATION – each replica of particle has its strategic parameters mutated
- REPRODUCTION - each mutated particle (plus the original one) generates an offspring through recombination, according to the particle movement rule, described below
- EVALUATION - each offspring has its fitness evaluated
- SELECTION – using a selection procedure (typically elitist selection), the best particles survive to form a new generation, composed of a selected descendant from every individual in the previous generation

The difference to classical PSO schemes is obvious: EPSO includes the selection operator inherited from evolutionary algorithms. Selection acts on the phenotype but the weights are selected along with and so the weights that led to a particle with a better value survive.

The recombination rule for EPSO becomes:

$$\mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + \mathbf{V}_i^{(k+1)}$$

$$\mathbf{V}_i^{(k+1)} = w_{i1}^{*}\mathbf{V}_i^{(k)} + w_{i2}^{*}(\mathbf{b}_i - \mathbf{X}_i) + w_{i3}^{*}\mathbf{P}(\mathbf{b}_G^{*} - \mathbf{X}_i)$$

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

$\mathbf{b}_G$ – 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

w_{i2} – weight conditioning the *memory* term

w_{i3} – weight conditioning the *cooperation* or *information exchange* term

P – communication factor: a diagonal matrix with elements of value 0 or 1 determined by a communication probability p (to be discussed in the following section).

The symbol * indicates that the parameters will undergo mutation through the scheme

$$w^{*} = w(1 + \sigma N(0,1))$$

with σ as the learning parameter and N(0,1) being the normal distribution.

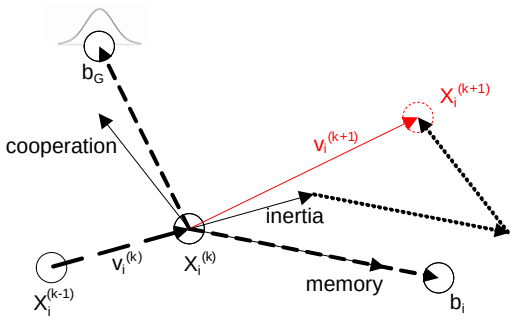

Figure 1 – Illustration of EPSO movement rule – a new individual is formed by recombination of four parents

In the most effective EPSO applications not only the weights affecting the components of movement rule are mutated but also the global best $\mathbf{b}_G$ is randomly disturbed to give $\mathbf{b}_G^* = \mathbf{b}_G + w_{i4}^* N(0,1)$ which has been demonstrated to help particles to avoid getting "stuck" in cases when the cooperation term evolves to become dominating and the other terms evolve by fading out. We have also witnessed cases where this parameter seems to be disturbing and then the selection pressure forces it to fade away becoming virtually zero.

One cannot help noticing that two random numbers become multiplied in this formulation, when one has $w_{i3}^* \mathbf{b}_G^*$: these would be $w_{i3}^* w_{i4}^*$. This should not be disturbing because the formulation adopted gives insight to what happens: we are aiming at an adaptation of w_{i3} relative to a probabilistically defined neighborhood of $\mathbf{b}_G$. By keeping the two factors independent, we allow that selection acts on one, the other or both elements with different consequences:

1. if the cooperation factor is harming the progression towards the optimum, selection will tend to favor small w_{13} regardless of w_{i4} (particles with smaller w_{i3} will be more successful and will survive)
2. if the cooperation factor is useful in the search for the optimum, selection will favor large w_{i4} if the best is still far from the optimum and smaller w_{14} when near the optimum.

So far we have only extensively tested models with equal weights in all space dimensions but there is a considerable interest in researching the behavior of the algorithm when selection may act independently in each direction. Also, we have so far adopted a single value p to all space dimensions but having distinct p_j for each dimension j is an interesting possibility to be studied.

In EPSO, the weights are strategic parameters and therefore integral parts of an individual. They are subject to selection which forces these parameters towards an efficient set of values that guarantee or accelerate convergence.

Unlike in classical self-adaptive evolutionary algorithms, where the mutation operator is adaptive, in EPSO the *recombination* rule is adaptive. In other words EPSO considers the recombination operator, not the mutation operator, as responsible for adaptation to search space properties. By adapting the weights, EPSO performs self-adaptation of the proportions of contributions of parents to form offspring. The process of selection acts separately on the descendants of each particle. This way, it becomes a parallel process where the interaction among particles is assured by the recombination rule.

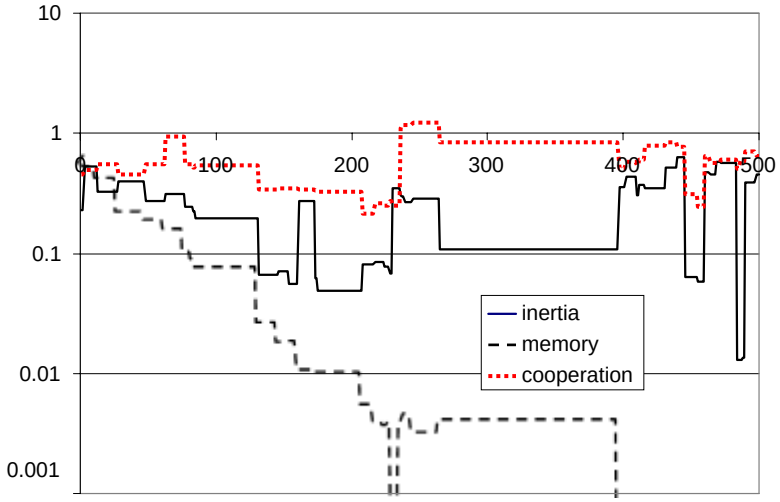

Figure 2 – Illustration of the process of evolving of EPSO weight values (y axis) over algorithm iterations (x axis), by representing the successive sets of weights for the particle that has discovered the current global best point in each iteration.

Figure 2 illustrates an example of weight adaptation during an optimization process - the weights from the best solution found in each iteration are shown on the graph. In this case, the selection procedure forced a self-adaptation of the memory weight causing it to fade away. This means that particles relying only in the inertia and cooperation terms to form their movement have proved stochastically to be more successful in progressing to the optimum than

competitors having larger influence of the memory term. The latter have been eliminated by selection. The values of all weights have had 0.5 as starting point in this case but we can also see that the selection has put the inertia term, most of the times, roughly one order of magnitude smaller than the cooperation term. This self-adapting characteristic has allowed EPSO to start with randomized weights in]0,1] instead of requiring initial special weights.

Stochastic Star communication topology

The classical, "full" star communication structure represents a communication scheme in which no particle has memory about the global best location, the particle $\mathbf{b}_G$ broadcasts its location in every iteration and all individuals receive this information at the same time.

In [11][12], alternative communication schemes, non-adaptive, single swarm, have been considered (other attempts exist that deviate from the scope of this paper). One of the conclusions from such work and the attempts of many other researchers in creating adaptive models is that a single static, unique and ideally universal communication scheme appropriate for all possible fitness landscapes does not exist, although some schemes have shown good performance on a range of problems. To address this problem, one may think of three ways: 1. to cluster problems according to specific characteristics allowing the establishment of a correspondence between problem classes and communication topologies desirable, and then to build an identification method able to classify a new problem into one of the clusters so that a specific fixed topology may be used, or 2. to avoid fixed communication topologies and to specify an adaptive mechanism derived for the observation on how the algorithm usually works, or 3. to build an intelligent approach that would in every moment self-adapt the communication topology to the type of problem being dealt with, avoiding a priori classifications. The latter is the approach followed by EPSO as much as possible, although we are still far from a fully self-adaptive method in this respect.

The basic EPSO equation includes communication factors **P**, thus introducing a *stochastic star* topology for the communication among particles. The communication factor P is a diagonal matrix containing binary variables of value 1 with probability p and value 0 with probability (1-p); the p value, externally fixed, controls the passage of information within the swarm and is 1 in classical formulations (the *star*). This means that, for each dimension in the search space and in each iteration, there is a probability p that a particle will not receive (use) the information available on the best location found by the swarm – the particle evolves then only under the influence of inertia and memory components or, in EA language, the recombination operator uses only 3 parents and discards the global best as a contributing parent for that specific dimension.

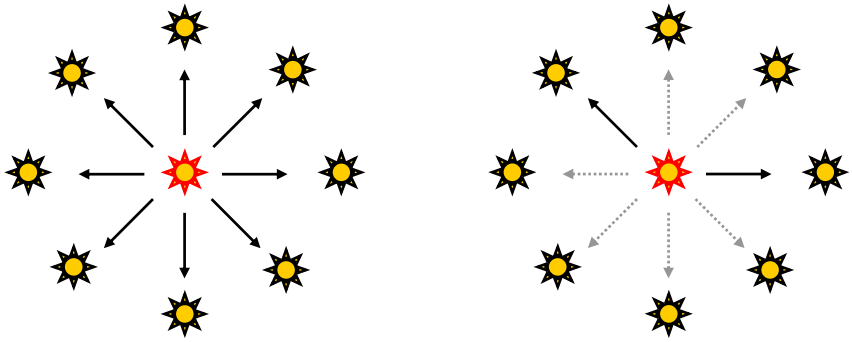

Figure 3 – Illustration of the star communication topology (left) and the stochastic star topology (right)

The stochastic scheme oscillates between the star arrangement and a selfish version called *cognitive model* in [13], where no communication exists and a descendent of an individual is built only of contributions from its ancestor line. One believes that restraining the free flow of information about the global best allows more local search by each particle, eliminates disturbing noise, allows the dynamics of particle movement to be more stable and avoids premature convergence.

Initial experiments with EPSO and the stochastic star concept, applied to a limited set of test functions and real life problems, and treating all dimensions in the same way (only one sampled random value compared to p per particle valid for all dimensions in each iteration, in a scheme that we call vector-stochastic) have misled us into

believing that a relatively low communication probability of p = 0.20 would usually lead to better results than a classical deterministic star model with p = 1. As we will show, treating each dimension independently (leads to even better results (a scheme that we call dimension-stochastic) and further research is needed.

Moreover, as the stochastic nature of communication is independently applied to each dimension of the global best, the stochastic star also represents unreliability in the communication channel. In other words, the information broadcasted in each iteration and in a particular dimension about the location of the global best could be lost or become corrupted with partial loss within the communication channel between particles.

It has been shown that a full star communication acts against exploration of the search space and may lead to premature convergence. On the other hand, alternative structures with extremely low communication between particles, like the ring structure where each particle informs only its direct neighbor, happen to affect the social component of particle swarm optimization and risk to convert the particle swarm search process into something close to a set of parallel individual searches.

A way for generating efficient communication topologies becomes desirable and justified. Having this in sight, a significant benefit from adopting a stochastic star communication has been verified. Its implementation is algorithmically simple: it is only necessary to implement a probability threshold below which communication is allowed, and above which the communication isn't allowed. Recent work by Clerc [14] has also given some theoretical observations on similarity of the stochastic generation of neighborhoods and stochastic star. The simplicity of stochastic star implementation suggests that it has the potential for a scheme for communication to be made adaptive in the future.

Effect of using different probabilities of communication

As said, initial experiments with stochastic star probability given as an external parameter to EPSO led to admit adopting the value of p = 0.2 as a rule of thumb [3], having the stochastic star probability constant during the optimization process. But extensive testing including in real-life applications suggested that this value might not be ideal for all cases. Confirming this assessment, the tests on several typical mathematical functions from the test suite in [15] are presented in this section. In all cases the population was set to 20 particles and the stopping criterion was a fixed number of fitness function evaluations. All others thing unchanged, including random number generator initialization, the tests were conducted by repeating the EPSO algorithm runs and varying only the communication probabilities. All weights were randomly initialized in the interval]0,1] (an indication that EPSO performance is not greatly dependent on initial weight setting). The learning rate has in all cases been set to 0.2.

Rosenbrock function

The Rosenbrock function has a global optimum difficult to reach, due to its twisted topology in the optimum vicinity. It serves as a test for convergence in difficult circumstances.

$$f_{Rosenbrock}(x) = \sum_{i=1}^{n} (100 \times (x_{i+1} - x_i^{\,2})^2 + (x_i - 1)^2) \text{ , tested on } [0, 30]^{30}$$

There were 20 runs for each test. The average error (relative to the known optimum) and RMSE (root mean square error) of the best solution from optimum are in the following Table 1.

Figure 4 shows the algorithm results after 50000 fitness function evaluations. It is clear that p = 0.2 isn't optimal – the most desirable value is around p = 0.75, which suggests that the optimization of the Rosenbrock function profits from limiting the communication among particles but not to a high degree. However, if the value of communication probability is further raised, the algorithm performance again deteriorates. Observe that when the error value is better also the *robustness* of the algorithm (defined as the capacity to converge to the same result in different runs) improves. The robustness of the algorithm is a concept in meta-heuristics of industrial application interest: in a real world environment where an operator has only the chance to run once the procedure before making a decision, the level of confidence that it will not produce solutions very different from one time to another, under the same circumstances, must be as high as possible. An algorithm with these characteristics has more value in practice, in these cases, than an algorithm that may give once an extremely good value and in another run a very bad value.

A way to measure the robustness of a meta-heuristic is to evaluate the amplitude of the dispersion of errors relative to the optimum by the RMSE. If the algorithm converges well to the optimum, the average error will in general correlate with the RMSE. This correlation will be disturbed if the solution gets stuck in local optima.

Table I – Rosenbrock function tests

	50.000 evaluations		100.000 evaluations	
communication probability p	average error	RMSE	average error	RMSE
0.1	10.505	6.693	8.691	4.992
0.2	10.354	10.078	9.855	6.537
0.3	9.923	6.088	6.909	4.111
0.4	9.179	6.116	4.356	3.5867
0.5	5.321	6.906	5.295	15.805
0.6	2.094	1.889	0.076	0.138
0.7	0.382	1.133	0.0061	0.0187
0.8	0.523	2.195	0.0002	0.0007
0.9	74.925	218.85	8.754	17.024
1.0	336.81	455.93	29.63	35.18

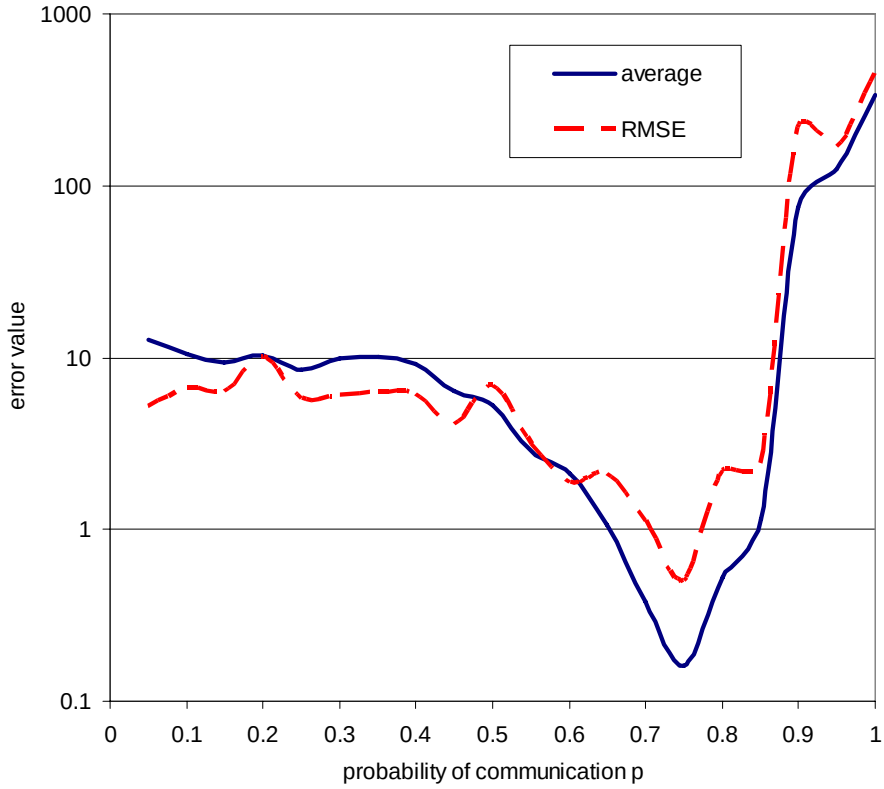

Figure 4 – Rosenbrock function after 50000 fitness function evaluations – average and RMSE of achieved error values for 20 runs. Y-axis scale is logarithmic.

In other words: in EPSO with stochastic star a better value of probability of communication positively affects

convergence *and* robustness. To confirm this indication, the same tests are also conducted with a less restrictive stopping criterion – with 100.000 fitness function evaluations and the results are depicted in Figure 5.

In comparison with the previous figure, several implications can be stated. Choosing sub-optimal values of communication probability p led to very weak improvement – the achieved error values for suboptimal regions of p [0, 0.5] and [0.85, 1] do not improve significantly after 50000 iterations.

On the contrary, when the p is inside interval [0.5, 0.85], closer to the optimal value, the algorithm's convergence is significantly better. The compliance of robustness and performance is again confirmed – *better* values of p led to *both* better robustness and better convergence properties.

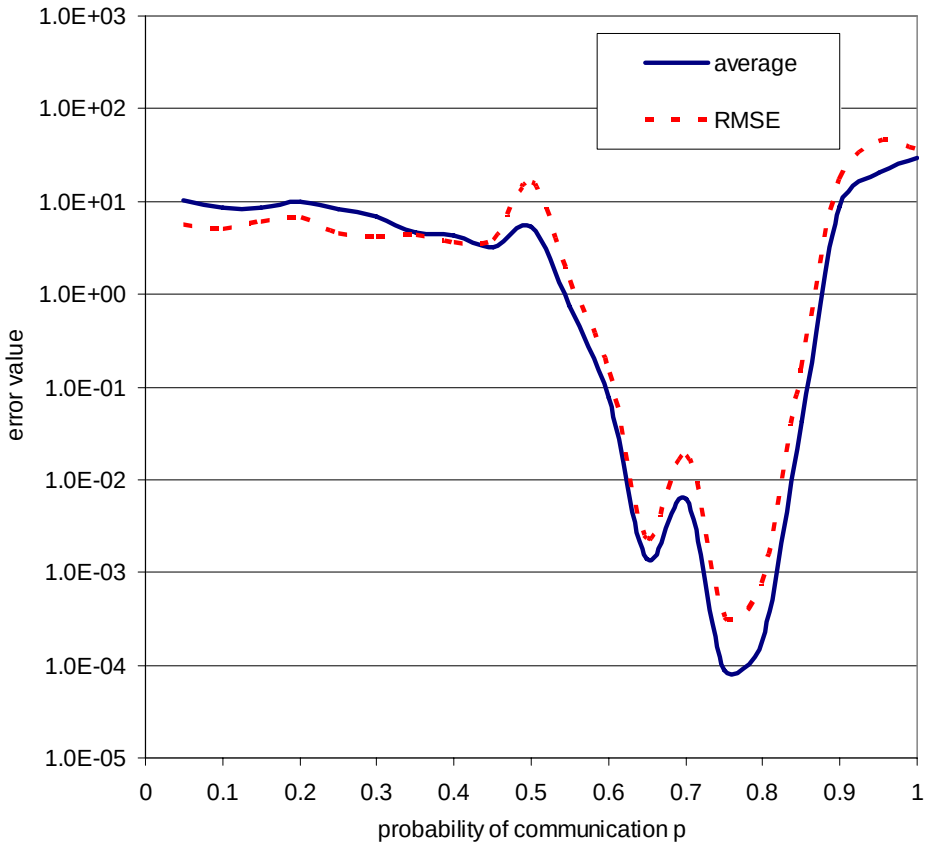

Figure 5 – Rosenbrock function after 100000 fitness function evaluations – average and RMSE of achieved error values for 20 runs. Y-axis scale is logarithmic.

CEC 2005 Shifted Schwefel's Problem

Another difficult test function is the 10-dimension Schwefel function from CEC 2005 [15] battery of test problems. This is a shifted, unimodal, non-separable and scalable function, tested on 10-dimensional hypercube [0, 100][10].

$$f_{Schwefel}(\mathbf{x}) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2 + f\_bias,\ \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, x_3 \ldots x_D]$$

As it is visible from Figure 6, with this function EPSO exhibits sub-optimal behavior with very low and very high

values of probability of communication. Outside these regions, the algorithm seems to be insensitive to altering the probability of communication.
An interesting characteristic is visible if the behavior of the algorithm is compared to the one with the Rosenbrock function. Even though EPSO with Schwefel's function shows very different responses when changing the probability of communication, the algorithm robustness again follows the algorithm performance. When the algorithm performs better – the average in a set of runs gets closer to the optimum, also the diversity of achieved values becomes lower.

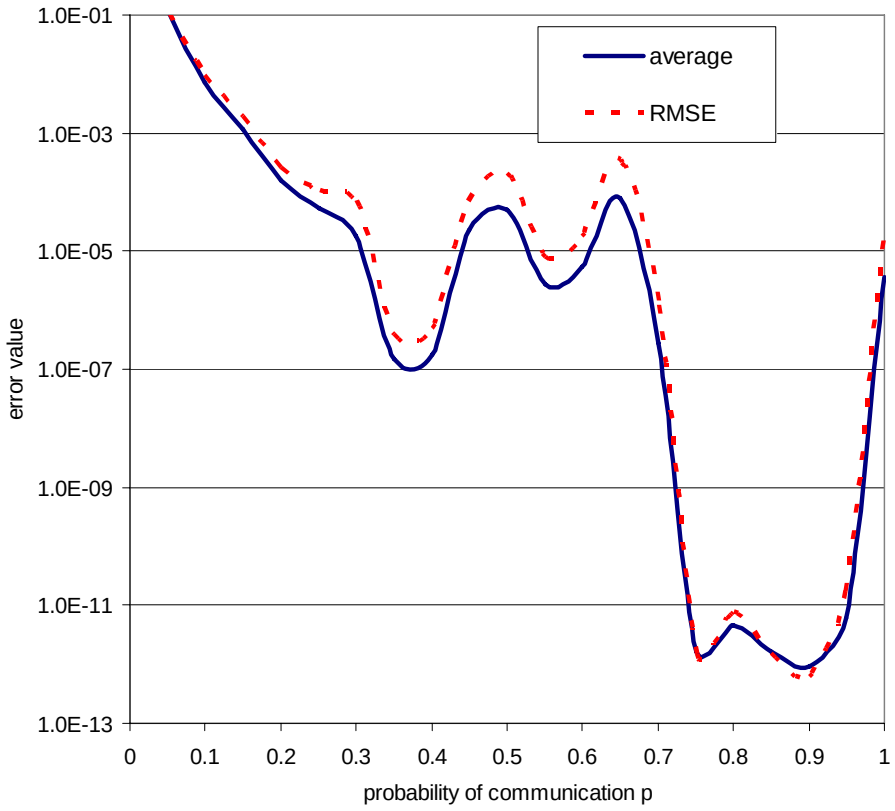

Figure 6 – Shifted Schwefel's function, runs stopped after 100000 fitness function evaluations – average and RMSE of best achieved error values for 20 runs. Y-axis scale is logarithmic.

CEC 2005 Shifted Rastrigin's Function

To illustrate various behaviors regarding communication probability, another function from CEC 2005 battery of tests is presented, shifted 10-dimensional Rastrigin's function:

$$f_{Rastrigin}(\mathbf{x}) = \sum_{i=1}^{D}\left(z_i^2 - 10\cos(2\pi z_i) + 10\right) + f\_bias, \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = \left[x_1, x_2, x_3 \ldots x_D\right]$$

It is a multimodal, shifted, separable and scalable function with particularly huge number of local optima, and its domain is restricted to hypercube $[-5, 5]^{10}$.

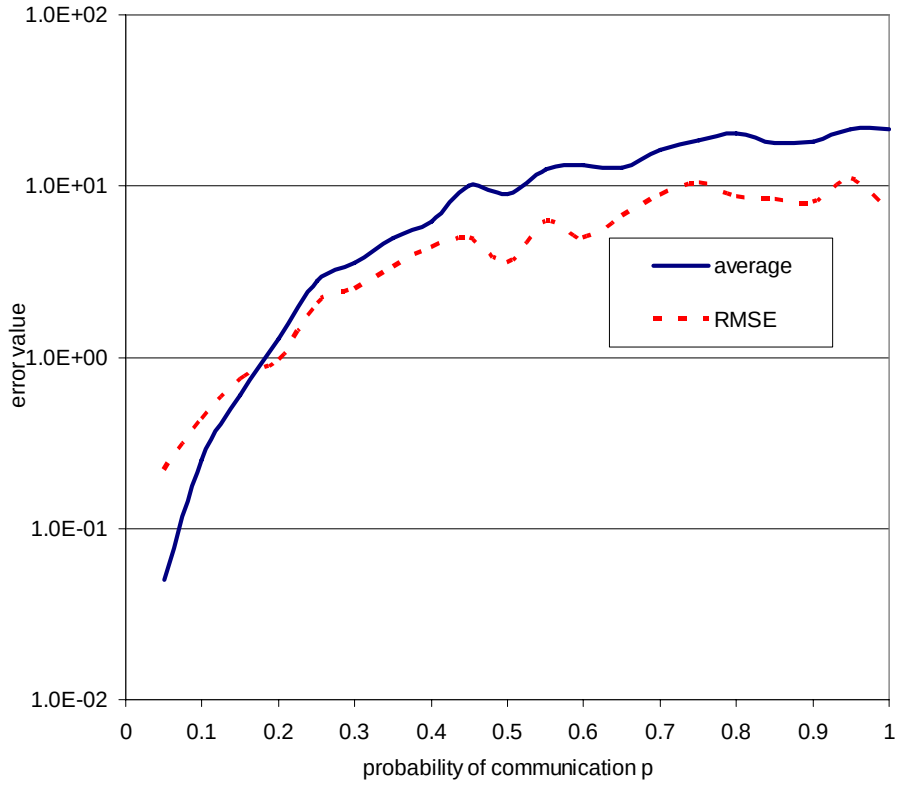

Figure 7 – Shifted Rastrigin's function, runs stopped after 50000 fitness function evaluations. Average and RMSE of best achieved error values for 20 runs. Y-axis scale is logarithmic.

On the contrary to previously examined functions, this function significantly profits from exploiting very low probability of communication. It has a relatively high number of local optima, so in this case parallel searches practically without exchange of information seem to be the best performers.

CEC 2005 Shifted Sphere Function

One of typical test functions used by evolutionary algorithms researchers, especially in theoretical observations, is Sphere (or Parabola) function. The function value basically represents Euclidean distance from the optimum. CEC 2005 version is unimodal, shifted, separable and scalable function in 10-dimensional space. The domain is restricted to hypercube [-100, 100][10].

$$f_{Parabola}(\mathbf{x}) = \sum_{i=1}^{D} z_i^2 + f\_bias, \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, x_3 ... x_D]$$

The graph represents runs with only 1000 fitness function evaluations. If allowed to run for higher number of fitness function evaluations, EPSO reaches the optimum practically regardless of probability of communication used.

However, the correlation in behavior of algorithm robustness and performance is again confirmed: for values of p where the algorithm has slower convergence, its robustness is also worse. This holds even for such low number of algorithm iterations and for a simple function being optimized.

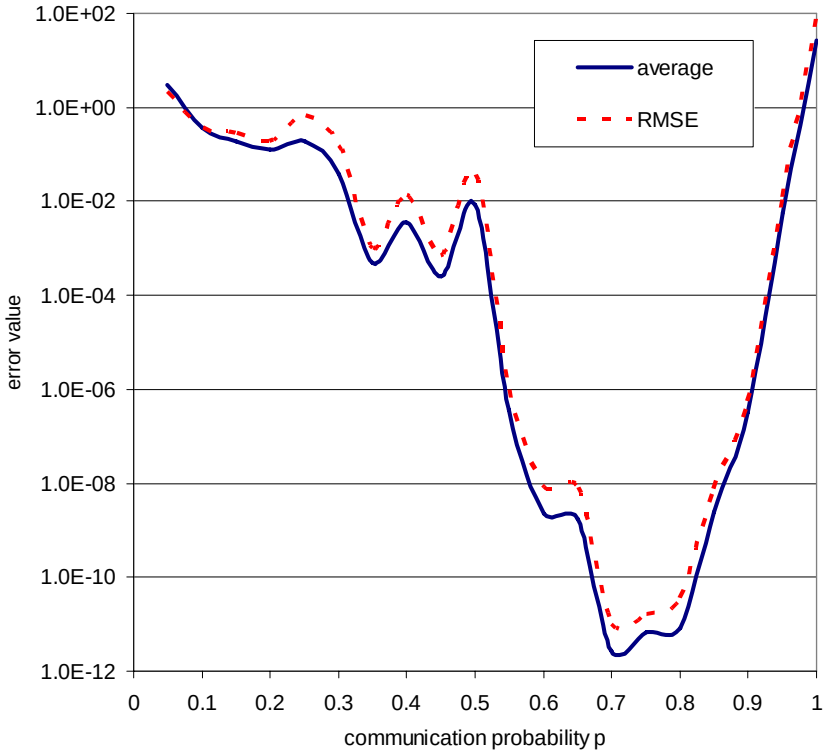

Figure 8 – Shifted sphere function. The optimization is stopped after only 1000 fitness function evaluations and RMSE of best achieved error values for 20 runs. Y-axis scale is logarithmic.

Application of EPSO to reactive power planning problem

In this section we present the results of the application of EPSO to an important practical problem in the domain of Power Systems. We understand that full apprehension of the problem is likely to be achieved only by people in the area. The following description has enough material to make it conceptually fully understandable to power system experts although we recognize that people outside the area may not be familiar with vocabulary or assumptions. However, we wish to make the case that the behavior observed in test problems is present in complex real world cases.

The reactive power planning (RPP) is a demanding problem in power system optimization. Reactive power is needed for the operation of a number of electric devices, but the flow of reactive power throughout the electric network is in many cases undesirable, because it originates power losses and loss of transmission capacity (with financial consequences) as well as unacceptable (low or high) nodal voltage levels. It is however possible to regulate the amount of reactive power injected in a power system – and reactive power compensation devices are used in that case.

RPP can be defined as determining the location and capacity of reactive power compensation devices to be purchased and installed within a given planning period. While trying to keep investment as low as possible over the planning period, the voltage profile (voltage levels throughout the power network) should be adequate and also energy losses in the network should be minimized. These criteria are clearly conflicting in many cases, so the optimal solution becomes depending on a tradeoff. Also, while the reactive power *control* problem gives only the optimal set of control variables [16], reactive power *planning* [17] defines the most appropriate locations to install the reactive power compensation devices and their types and sizes.

The problem has added difficulty since during the day or the year load levels in an electric power system vary significantly, so a set of control variables, perfectly suitable for daily periods with high network loads, could be highly inappropriate for night periods with lower loads. Control variables are responsible for keeping the voltage profile within adequate limits and avoid voltage collapse or overvoltages. Therefore, besides the amount of reactive power to be compensated by newly installed devices, the problem variables have also to include other variables such as transformer tap settings and settings on voltage-regulated buses (with generators, switched capacitor banks or static compensators).

To be able to assess the network conditions and determine whether the operating conditions are acceptable, load flow calculations are required for every set of problem variables. This makes the fitness function, responsible for the evaluation of solutions, significantly heavier in computational terms than the mathematical test functions. Its complexity derives not only from this fact but also from being associated with discrete or non-differentiable cost functions.

By taking advantage of EPSO robustness of convergence, we were able to build an application that proposes decisions for future installation of reactive power compensation devices. EPSO was chosen since it has already proven to be successful in solving similar problems from the domain of power system operation [2], [18], [19]. The distinctive characteristic of new application is that it is able to take into account not only a basic scenario, but also selected contingency scenarios associated with given probabilities. The multiple scenario approach makes the calculation of fitness even more difficult in computational terms.

There are two factors that distinguish network scenarios: load levels and network structural changes. Structural scenarios are distinct from one another in the fact that they correspond to a diversity of contingencies that might occur in the network. Contingencies in power systems are related to discrete events or situations where network elements become unavailable with direct influence on network performance. Each contingency case is associated with a given probability or unavailability, and usually one considers a base case with a high probability value and then contingency scenarios. Different load levels correspond to different energy/power demand.

Finally, each scenario in the RPP corresponds to a combination of load and structural scenarios and its probability will be the product of partial scenario probabilities. EPSO fitness function for this problem is built from three different functions: capacitor investment cost, cost of energy losses and penalization function for violating voltage limits. Notice that we use here the terms *cost* and *fitness* as equivalents although some people prefer to define fitness as a maximizing function: if cost is an index of fitness, when one is minimizing cost one is maximizing fitness.

This fitness function is subject to constraints related to a reactive planning problem. The problem is basically formulated as:

$$\text{Minimize } OBJ = \sum_{s=1..N_s} \Pr_s \cdot \left(CC + EC + VL\right)_s$$

$$\text{subject to } \forall s \begin{cases} f(P,Q,V,\theta,T) = 0 \\ Q_{ci}^{\min} \le Q_{ci} \le Q_{ci}^{\max}, i \in N_c \\ Q_{gi}^{\min} \le Q_{gi} \le Q_{gi}^{\max}, i \in N_g \\ T_i^{\min} \le T_i \le T_i^{\max}, i \in N_T \\ V_i^{\min} \le V_i \le V_i^{\max}, i \in N_B \end{cases}$$

where

s = index of network scenarios

$\Pr_s$ = probability of scenario s

CC = capacitor investment function

EC = energy cost function

VL = voltage limit penalty function

and

$f(P,Q,V,\theta,T) = 0$ - equality constraints of power flow problem, as a function of power injections (active P and reactive Q), nodal voltages (magnitude V and angle θ) and transformer taps T

Q_{ci} = reactive power source installation at node *i*

N_c = set of candidate nodes for reactive power installation
Q_{gi} = reactive power generation at node *i*
N_g = set of generator nodes
T_i = transformer tap setting – the transformer with tap changers on winding connected to bus *i*
N_g = set of nodes with transformer windings with tap changers
V_i = voltage magnitude at bus *i*
N_B = set of buses

The conditions stated above must hold for each observed load level. Investment costs for each reactive power injection device consist of installation cost and purchase cost:

$$\mathrm{CC} = sf_C \left[\sum_{k \in N_C} \mathrm{C}_k = \sum_{k \in N_A} (\mathrm{C}_{i,k} + \mathrm{C}_{p,k}) \right]$$

where sf_C represents a scale factor of the investment function, N_C represents candidate buses, N_A the set of nodes where the devices may be installed, a subset of the set of candidate nodes. $\mathrm{C}_{i,k}$ and $\mathrm{C}_{p,k}$ represent the cost of installation and purchase of devices for *k*-th candidate bus, respectively.

The algorithm is capable of per-node handling of different installation and purchase costs, in order to take into account variations related to specific installation conditions. For instance, the price for installation of a compensation device in a city centre might be higher than in a rural area and the actual installation costs have to include the penalties needed to be paid if energy is not supplied to customers. The devices that already exist in the network can get zero cost and may be either reinforced or ignored.

To be able to cope with varying network load levels, more expensive switched capacitor banks can be used in place of fixed capacities. The algorithm handles purchase costs for each bank by breaking it down in two components:

$$\mathrm{C}_{p,i} = NFC_i \cdot CF_i + NSC_i \cdot CS_i$$

where, for node *i*, NFC_i represents quantity of fixed capacitor banks, CF_i per-unit cost of fixed capacitors, NSC_i quantity of fixed capacitor banks, and finally CS_i per-unit cost of switched capacitors.

The second part of fitness function takes into account total power losses in the system are used as main factor for energy loss function. For each load level and structural scenario observed, the system losses are calculated from Newton-Raphson power flow results and the load level duration. The formulation is

$$EC = sf_E \left[\sum_{j=1}^{N_L} PL_j \cdot T_j \cdot MC_j \right]$$

where

sf_E represents scale factor for energy cost function,
N_L number of load levels,
PL_j power loss at load level *j*,
T_j planning time duration of load level *j*, and
MC_j marginal cost of energy for load level *j*.

The costs are specified for each load level since marginal energy prices usually have high volatility and significantly vary between peak and off-peak periods. The total voltage level penalty is a linear combination of factors proportional with linear and squared voltage deviation, which make possible to represent penalty policies. The basic idea is to make this constraint “soft”, in order to avoid impairment of exploration of search space by too strict restrictions.

$$\mathrm{VL} = sf_V \left[\sum_{j=1}^{N_L} \left(\sum_{i \in N_B} SF_{i,j} + sf_M \sum_{i \in N_B} LF_{i,j} \right) \right]$$

where

sf_V is scale factor of voltage limit penalty function, and
sf_M represents scale factor for linear penalty function.

The solution is penalized if its node voltages are out of limits. Penalties thus distort the search space and influence EPSO particle movement with a gradient that pushes particles out of regions with undesirable voltage conditions.

Large penalty factors ensure that the algorithm always prefers solutions with acceptable voltage levels, so the particles are vigorously kept in the regions with acceptable voltages – but they do not allow the discovery of a solution close to the boundary that would lead to a significant reduction in investment. There is also a trade-off here. The final planning application requires the following inputs: a network configuration (i.e. nodes and branches of a network configuration, for base and for contingency cases), subset of nodes defined as candidate nodes for the installation of devices, sets defining available transformer tap settings, and finally node voltage limits for nodes with voltage regulation – plus all costs.

The variables are encoded in a search vector space as follows: the first N_c variables represent the reactive power injection in a particular node, ranging from $Q_{ci}^{\min}$ to $Q_{ci}^{\max}$. Each value Q_{ci} determines the amount of capacity needed at node *i*. N_T variables represent settings of transformer taps, ranging from $T_i^{\min}$ to $T_i^{\max}$ and finally N_{VR} variables represent N_{VR} settings for voltage-regulated nodes, ranging from minimum to maximum voltages for each node.

$$X = [Q_{c,1}^1 \ldots Q_{c,N_C}^1, T_1^1 \ldots T_{N_T}^1, V_1^1 \ldots V_{N_{VR}}^1 \ldots Q_{c,1}^{N_L} \ldots Q_{c,N_C}^{N_L}, T_1^{N_L} \ldots T_{N_T}^{N_L}, V_1^{N_L} \ldots V_{N_{VR}}^{N_L}]$$

So far the number of vector space dimensions is $N_c + N_T + N_{VR}$. These variables are further repeated N_s times in order to represent the situation for each of load levels at each structural configuration scenario.

Although investment depends on the capacitors to be installed, and these are derived from the *Q* variables, this algorithm takes in account the margin of control available in the system (using transformer taps and generator excitation) and therefore avoids possible excessive investment. For each scenario a Newton-Raphson power flow calculation is performed, energy losses and node voltages follow.

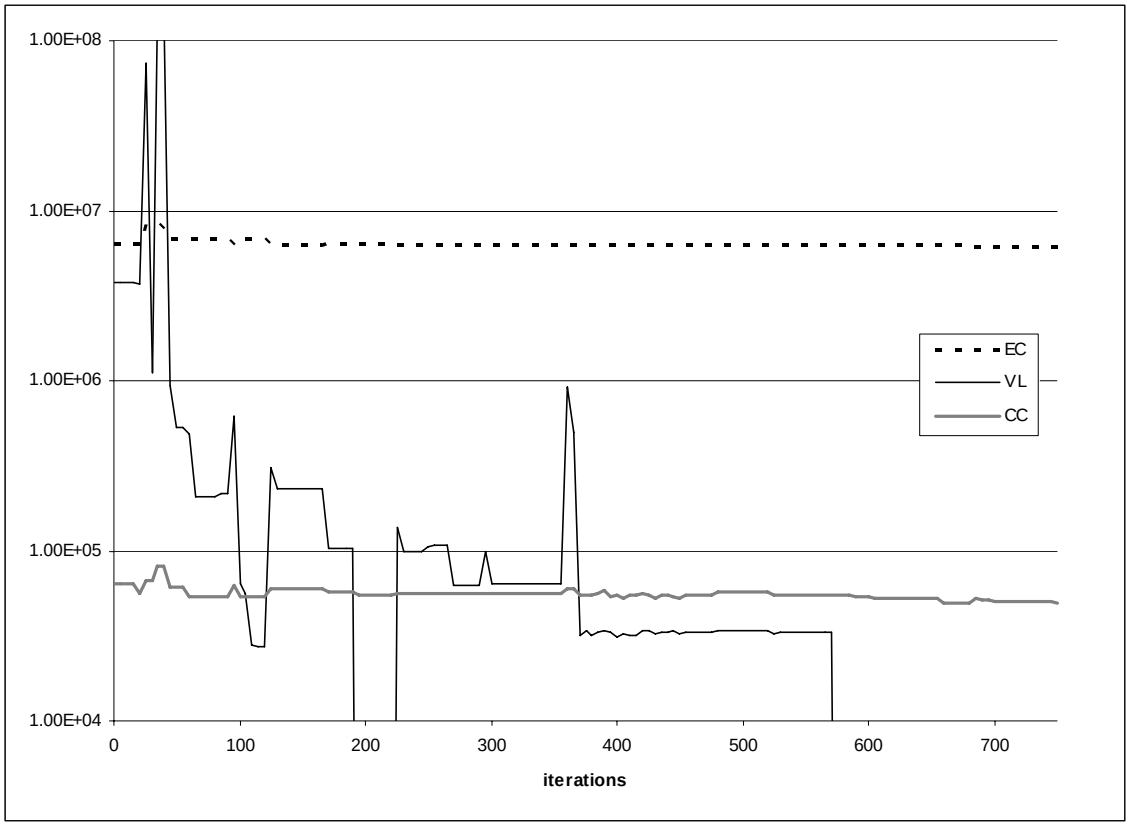

Figure 9. An illustration of the evolution of cost components through algorithm convergence for IEEE 30 bus test case; voltage deviations higher than 3% are penalized. In this case, voltage control was critical and while only small variations in losses or investment were observed, the algorithm was seeking a configuration that would eliminate voltage penalties.

When all scenarios are examined, new device sizes and types are determined. If the necessary reactive power injection for a candidate node is the same over all periods, then a cheaper fixed injection device (a fixed capacitor bank) should be purchased. Otherwise, if the power injection in a candidate node varies over periods, a fixed size device needs to be purchased for the lowest determined reactive power needed, and more expensive switched or varying injection devices for the remainder of injected power should also be purchased.

The algorithm has been successfully implemented using a high-performance computation library developed in INESC Porto. It has been tested on both simulation cases and real networks. A case of testing the algorithm on IEEE 30 node test system is presented. The full set of data may be requested from the authors,

To illustrate the trade-off that a planner must face in finding the optimal solution for a particular case, the IEEE 30 node network has been optimized with different values of allowed voltage deviations. Planner allowing higher voltage deviations, in other words allowing the operating conditions in the network to be farther from "ideal", can achieve higher energy savings while keeping the investments lower. This is illustrated in the following table:

max. allowed voltage deviation	initial	>3%	>5%	>8%
generated power P_g[MW]	161.72	160.85	160.76	160.63
total network losses P_{losses} [MW]	3.25	2.35	2.26	2.13
maximum voltage deviation in final solution, %	-4.35	+3.00	+4.91	+7.7%
installed capacitor banks [MVAr]	-	58.1	56.2	50.8

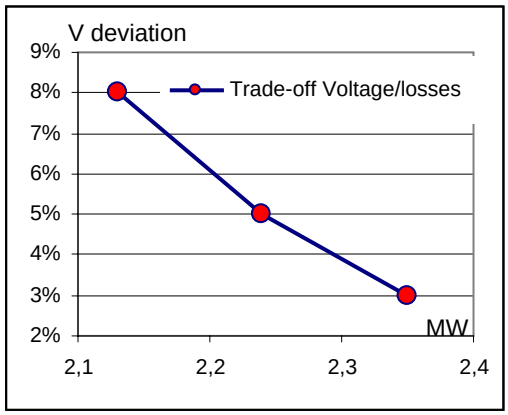

Figure 10. Admissible voltage band vs. power losses in the IEEE 30-bus system: a Pareto optimum front

According to the specific company policy, a planner is responsible for choosing a point on a Pareto front such as in Figure 10. Further development of this application will be towards employing techniques that minimize the planner's effort: a foreseen enhancement is to couple the algorithm with Monte Carlo simulations, which will become responsible for generating scenarios (network states) which is particularly important in today's distribution networks containing distributed generation. This is made possible due to the acceptable running times of the EPSO algorithm. The EPSO employed in the application for reactive power planning also takes advantage of the stochastic star communication scheme.

Figure 11 shows the variation of the total costs of the best solution found by EPSO for reactive power planning on the IEEE 30-node test network in 5 runs for each p value. The communication probability p is the only parameter being systematically modified, the weights being randomly initialized in]0,1]. The results reveal that altering the communication probability does not severely affect algorithm convergence – the maximum difference in costs compared is below 2% after 200 iterations. However, the advantage of not having full communication is still observed, namely in the fact that at lower p values the quality of convergence is assured for a much smaller computing effort.

Also, if the algorithm is allowed to run longer (more iterations), the difference between optimal and suboptimal

values of p further diminishes. Note, however, that the behavior of the RPP function is very dependant on the network configuration and the effect of a stochastic star in the quality of convergence may become relevant in problems of a larger scale than the one discussed above.

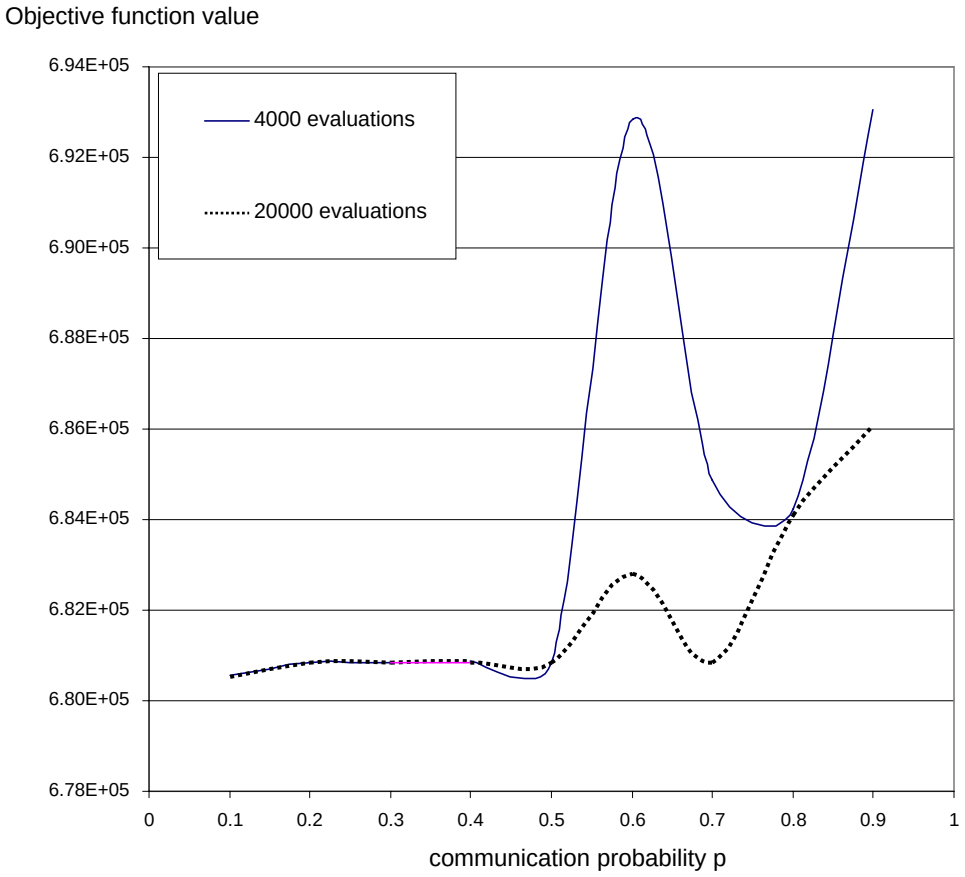

Figure 11. Results of EPSO for the RPP in relation to the stochastic star probability tested on IEEE 30-node network with varying communication probability (average of five runs).

Conclusion

Both from theory and experiments, various researchers have confirmed that the "social feature" of particle swarm algorithms – scheme of interchanging information between particles – is of crucial importance for its performance. Static communication schemes, initialized only when starting the optimization process, also seem not to be the desired choice, a conclusion not only derived from the results with p = 1 in this paper but also from the comparisons reported elsewhere [3] between the performance of EPSO and PSO. The paper presents a stochastic star scheme that implements an oscillating social network topology – the one that alternates between selfish-ignorant and fully cooperative scheme. One remarkable advantage lies in its algorithmically simple implementation and no need for specific weights (random initialization seems enough).

Tests conducted with the external setting of stochastic star probability indicated a correlation between algorithm robustness and its performance. They are strongly correlated – when the stochastic star probability (and hence the social network topology) is optimal for the particular fitness landscape, the algorithm's performance and robustness

both improve at the same time. The algorithm reaches the optimum with less fitness function evaluations and is also more reliable, i.e. has less divergence among results achieved in repeated runs. This is a result expected in general and should especially be expected for an algorithm not getting stuck in local optima.

This conclusion holds for both difficult and simple mathematical functions. Unfortunately there is no "typical" behavior and hence no actual optimal value of communication probability that would be suitable for all functions could be found. It seems as if the algorithm performance would require more local search in problems with multiple local optima and this local search is achieved with low values of p. The problem here is again of recognition a priori of the type of problem one is facing, in order to select an adequate p (if p = 0.1 then the particle, in a specific dimension, will only use information on the whereabouts of the global best once in ten iterations on average, meaning that its behavior will be dominated by the memory factor most of the iterations – or, in other terms, only one out of every ten dimensions of a particle receive information about the location of the global best in each iteration) . This observation leads the authors to strongly believe on the potential and feasibility of an adaptive (or even self-adaptive) scheme for the variable of p as well as distinct p values for every space dimension – this would eliminate the need for an a priori recognition and instead would learn from examples during the solving of the problem.

This dimension-stochastic version, where allowing information to reach a particle is sampled on a per dimension basis, has proven in our experiments to be more efficient that earlier versions where the decision to receive information about the global best was made on a per particle basis, even if governed by a probability threshold. This made us review the concept of communication probability and to step back on recommended values earlier published. It is clear that our version of EPSO seems to behave more efficiently that the PSO code made public in [4] – a result not reported in this paper but in a related publication [3] – but it is also clear that more research is required to make the progress mechanisms of EPSO with stochastic star fully understood.

The paper also presents a successful application of EPSO to the reactive power planning problem, a difficult problem from the domain of power system optimization, with a complex objective function and a large scale search space – in a real world problem, this could easily reach the scale of hundreds of variables. EPSO performance in this problem also displayed the behavior detected in laboratory test functions, namely showing that a stochastic star structure speeds up considerably the solving of the problem.

Further research will be directed towards investigating various particle "communication skills", coupled with creating adaptive stochastic communication schemes based on the principle of the stochastic star. The simplicity of the stochastic star scheme and the simplicity of its updating will be of helpful nature in these efforts.

References

[1] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *Proceedings of IEEE International Conference on Neural Networks (ICNN'95)*, Vol. IV, pp.1942-1948, Perth, Australia, 1995.

[2] V. Miranda and N. Fonseca, "EPSO – Best-of-Two-Worlds Meta-Heuristic Applied to Power System Problems", *Proceedings of WCCI/CEC – World Conference on Computational Intelligence, Conference on Evolutionary Computation*, Honolulu (Hawaii), USA, May 2002.

[3] V. Miranda, H. Keko and A. Jaramillo, "EPSO: Evolutionary Particle Swarms", Ch. 6 in "*Advances in Evolutionary Computing for System design*", L. Jain, V. Palade, D. Srinivasan Eds., Springer, series: Studies In Computational Intelligence, ISBN 978-3-540-72376-9, Volume 66, pp. 139-168, 2007.

[4] M. Clerc et al., Standard PSO 2006, in Particle Swarm Central, http://www.particleswarm.info/, code retrieved Dec. 2006.

[5] V. Miranda, "Evolutionary Algorithms with Particle Swarm Movements", Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems - ISAP 2005, pp. 6-21, Arlington (VA), USA, 6-10 Nov. 2005

[6] V. Miranda and N. W. Oo, "New experiments with EPSO – Evolutionary Particle Swarm Optimization", *Proceedings of IEEE Swarm Intelligence Symposium 2006*, Indianapolis (Indiana), USA, May 2006

[7] H.-P. Schwefel, "Adaptive Mechanismen in der biologischen Evolution und ihr Einfluß auf die Evolutionsgeschwindigkeit", *Technical report*, Technical University Berlin, 1974

[8] D.B.Fogel, "Evolving Artificial Intelligence", Ph.D. Thesis, University of California, San Diego, 1992

[9] H.-G. Beyer, “Toward a Theory of Evolution Strategies: Self-Adaptation”, in *Evolutionary Computation*, vol. 3, no. 3, pp. 311-347, 1996

[10] S. Meyer-Nieberg and H.-G. Beyer, “Self-adaptation in evolutionary algorithms” in F. Lobo, C. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithm*s, Springer, series: Studies in Computational Intelligence , Vol. 54, 2007

[11] R. Mendes, “Population Topologies and Their Influence in Particle Swarm Performance”, PhD Thesis, University of Minho, Portugal, April, 2004

[12] R. Mendes, J. Kennedy and J. Neves, “The fully informed particle swarm: simpler, maybe better”, *Evolutionary Computation, IEEE Transactions on*, Vol.8, Iss.3, June 2004, pp. 204- 210

[13] J. Kennedy, “The Particle Swarm: Social Adaptation of Knowledge”, *Proceedings of the 1997 International Conference on Evolutionary Computation*, pp. 303-308, IEEE Press, 1997

[14] M. Clerc, “Back to random topology”, technical report, March 2007, available online at http://clerc.maurice.free.fr/pso/

[15] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, “Problem Definitions and Evaluation Critera for the Special Session on Real-Parameter Optimization”, Technical Report, Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, IIT Kanpur, May 2005.

[16] J.G. Vlachogiannis and K.Y. Lee, “Reactive power control based on particle swarm multi-objective optimization”, *Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on*, Vol., Iss., 6-10 Nov. 2005

[17] J.T. Ma and L.L. Lai, “Evolutionary programming approach to reactive power planning”, *IEE Proceedings on Generation, Transmission and Distribution*, Vol.143, Iss.4, Jul 1996, Pages:365-370

[18] W. Lou Chin, L.M. Proença, V. Miranda, “Demonstrating an Efficient Capacitor Location and Sizing Method for Distribution Systems - Application to the Macau Network”, Proceedings of *ELAB'96 - 3º Encontro Luso-Afro-Brasileiro de Planeamento e Exploração de Redes de Energia*, vol.1, ed. INESC, Porto, Portugal, Oct. 1996.

[19] V. Miranda and N. Fonseca, “EPSO - Evolutionary Particle Swarm Optimization, a New Algorithm with Applications in Power Systems”, *Proceedings of IEEE T&D AsiaPacific 2002 - IEEE/PES Transmission and Distribution Conference and Exhibition 2002*: Asia Pacific, vol.2, October 2002, pp.745-750.