INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrotécnica Energia e Automação

Particle Swarm Optimization-based Algorithm for Optimal Reactive Power Dispatch

Ruben Henrique de Brito Azancot de Menezes

(Licenciado em Engenharia Eletrotécnica)

Dissertação para a obtenção do grau de Mestre em Engenharia Eletrotécnica – ramo de Energia

Orientadores:

Professor Pedro Miguel Neves da Fonte

Professor Rui José Oliveira Nóbrega Pestana

Júri:

Presidente: Professor Luis Redondo

Vogais:

Professor Francisco Reis

Professor Pedro Fonte

Junho de 2022

ABSTRACT

Reactive power optimization, more specifically the Optimal Reactive Power Dispatch (ORPD), is very important for the security and economy of power systems. It is a mixed integer non-linear optimization problem for which metaheuristic methods have proven to be effective in its solution. These methods do not guarantee that a global optimum is found, as premature convergence to local optima may occur. Much research is focused on improving the ability of metaheuristics in finding the global optimum. This thesis presents a particle swarm optimization-based algorithm (PSO) for the solution of the ORPD. The implementation of the PSO algorithm is explained in detail, including the parameter selection, a constraint handling method and a discrete variable rounding method. Another version of the PSO, called Fitness-Distance Ratio PSO (FDR-PSO), is implemented in order to improve the results of the basic PSO, by decreasing the chance of premature convergence to local optima. One other version of the PSO called Second Order PSO (SO-PSO) is implemented for the same purpose. The SO-PSO until this point remained somewhat untested in the ORPD. This second-order principle was combined with the FDR-PSO, making it the SO-FDR-PSO.

The implemented versions of the PSO were tested on the IEEE 14 bus and IEEE 39 bus systems with 30 runs each. On the 14 bus system, the lowest active losses found were of 12,280 MW. The basic PSO had an average of 12,387 MW showing that it often converged to good local optimal solutions, and occasionally 12,280 MW, which is possibly the global optimum (the best optimal solution the PSO was able to find). The solutions found were also always feasible, meaning the ORPD was effectively solved. The FDR-PSO was able to improve the results, with an average of 12,297 MW. The SO-PSO improved the results further, the most notable improvement being in the success rate, now of 57%, which means it often found solutions close to 12,280 MW, the apparent global optimum of the problem, and was observed to never converge prematurely to local optima. The SO-FDR-PSO had an average of 12,280 MW and success rate, of 100%, meaning the apparent global optimum was consistently reached. On the 39 bus system the overall results worsened in statistical terms in relation to the 14 bus system, although the solutions remained always feasible, still with a considerable decrease in active losses and the FDR-PSO/SO-FDR-PSO maintained its superiority over the PSO/SO-PSO.

RESUMO

A otimização de potência reativa, mais especificamente o Despacho Ótimo de Potência Reativa (ORPD), é muito importante para a segurança e economia dos sistemas de potência. Trata-se de um problema de otimização não linear inteiro misto para o qual métodos metaheurísticos têm se mostrado eficazes em sua solução. Esses métodos não garantem que um ótimo global seja encontrado, pois pode ocorrer convergência prematura para ótimos locais. Muitas pesquisas estão focadas em melhorar a capacidade das metaheurísticas em encontrar o ótimo global. Esta tese apresenta um algoritmo baseado em Otimização por Enxame de Partículas (PSO) para a solução do ORPD. A implementação do algoritmo PSO é explicada detalhadamente, incluindo a seleção de parâmetros, um método de tratamento de restrições e um método de arredondamento de variáveis discretas. Outra versão do PSO, chamada *Fitness-Distance Ratio PSO* (FDR-PSO), é implementada para melhorar os resultados do PSO básico, diminuindo a chance de convergência prematura para ótimos locais. Uma outra versão do PSO chamada PSO de Segunda Ordem (SO-PSO) é implementada para o mesmo propósito. O SO-PSO até este ponto permaneceu pouco testado no ORPD. Este princípio de segunda ordem foi combinado com o FDR-PSO, tornando-se o SO-FDR-PSO.

As versões implementadas do PSO foram testadas nas redes IEEE de 14 e 39 barramentos com 30 corridas cada. No sistema de 14 barramentos, as menores perdas ativas encontradas foram de 12.280 MW. O PSO básico teve uma média de 12.387 MW mostrando que muitas vezes convergiu para boas soluções ótimas locais, e ocasionalmente 12.280 MW, que é possivelmente o ótimo global (a melhor solução ótima que o PSO conseguiu encontrar). As soluções encontradas também foram sempre admissíveis, significando que o ORPD foi efetivamente resolvido. O FDR-PSO foi capaz de melhorar os resultados, com média de 12.297 MW. O SO-PSO melhorou ainda mais os resultados, sendo a melhoria mais notável a taxa de sucesso, agora de 57%, o que significa que frequentemente encontrou soluções próximas a 12.280 MW, o aparente ótimo global do problema, e observou-se que nunca convergiu prematuramente para ótimos locais. O SO-FDR-PSO teve uma média de 12.280 MW e taxa de sucesso de 100%, o que significa que o ótimo global aparente foi alcançado de forma consistente. Na rede de 39 barramentos os resultados no geral pioraram em termos estatísticos em relação à rede de 14 barramentos, embora as soluções encontradas tenham permanecido sempre admissíveis, ainda com uma diminuição apreciável das perdas ativas e o FDR-PSO/SO-FDR-PSO manteve sua superioridade sobre o PSO/SO-PSO.

KEYWORDS

Particle swarm optimization, fitness-distance ratio PSO, second order PSO, optimal reactive power dispatch, power loss minimization, nonlinear programming,

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor Pedro Fonte and Professor Rui Pestana, my thesis supervisors, for their patient guidance, suggestions, and critiques of this thesis. Their expertise in their respective areas was invaluable. I extend my thanks to all the other professors of the electrical engineering department of *Instituto Superior de Engenharia de Lisboa* for all I was thought during my academic path.

I would also like to thank my parents, brother, and other close family members for their encouragement and for always being there for me.

Last but not least, all researchers/engineers in the field of power engineering and mathematical optimization who indirectly contributed to this thesis.

LIST OF SYMBOLS [S.I UNIT]

B – susceptance [S]

c_1 – cognitive acceleration factor

c_2 – social acceleration factor

$f(X)$ – objective function value (active power losses) of particle with position X

G – conductance [S]

gBest – global best particle

$G_i(Y_j)$ – violation of constraint i by state variable vector Y_j

$minlim_{x_i^d}$ – minimum/maximum variation limit of position of particle i at dimension d

nBest – neighborhood best particle

pBest – individual/previous best particle

P_g – global best position

p_g^d – global best position at dimension d

P_{n_i} – neighborhood/local best position of particle i

$p_{n_i}^d$ – neighborhood/local best position of particle i at dimension d

P_i – individual/previous best position of particle i

p_i^d – individual/previous best position of particle i at dimension d

P_{loss} – active power losses [MW]

Q_C – shunt capacitor reactive power [Mvar]

Q_G – generator reactive power [Mvar]

S_l – line apparent power [MVA]

T_k – transformer tap ratio [p.u]

v_i^d – velocity of particle i at dimension d

v_{max} – maximum/minimum velocity of particles

v_{max}^d – maximum/minimum velocity of particles at dimension d

V_G − generator scheduled voltage [p.u]

V_i − velocity of particle *i*

X_i − position of particle *i*

x_i^d − position of particle *i* at dimension *d*

x_{max} − maximum/minimum position of particles

x_{max}^d − maximum/minimum position of particles at dimension *d*

Y_j − state variable vector for particle *j*

y_j^d − state variable vector for particle *j* at dimension *d*

y_{max}^d − maximum/minimum value of state variable vector at dimension *d*

δ – load angle [deg]

χ − constriction factor

$v(Y_j)$ − overall constraint violation by state variable vector Y_j

ω − inertia weight

ABBREVIATIONS

FDR-PSO – Fitness-Distance Ratio Particle Swarm Optimization

IEEE – Institute of Electrical and Electronics Engineers

NLP – nonlinear programming

ORPD – Optimal Reactive Power Dispatch

PSO – Particle Swarm Optimization

SO-FDR-PSO – Second Order Fitness-Distance Ratio Particle Swarm Optimization

SO-PSO – Second Order Particle Swarm Optimization

INDEX

INDEX OF FIGURES

INDEX OF TABLES

Chapter 1 – INTRODUCTION

1.1 – Motivation

Optimal Reactive Power Dispatch (ORPD) is a subproblem of optimal power flow calculation. Reactive power plays an important role in the voltage stability and active power transfer in a power system. Therefore, evaluation of the reactive power dispatch is important for the improvement of the security and operating costs of power systems. The objectives of the ORPD are: minimization of active power losses; improvement of voltage profile; minimization of transmission costs and maximization of voltage stability in the system. This is achieved by proper adjustment of parameters of the power grid, mainly location and size of shunt capacitors/reactors, tap ratios of transformers and the reactive power output of generators, improving the quality of the grid's voltage profile and reducing power losses.

Since the reactive power output of shunt capacitors and transformer tap ratios are discrete variables and the reference voltage (reactive power output) of generators are continuous variables, the ORPD problem is formulated as a mixed integer nonlinear optimization problem with both discrete and continuous variables. Also, the ORPD problem is non-convex and multimodal.

In the past, conventional optimization methods were proposed in literature to solve this problem but proved to find unsatisfactory solutions. For these reasons most methods employed recently to solve this problem are meta-heuristic. Metaheuristic methods have been shown to be effective in finding good solutions to complex problems such as the ORPD. Even though they are effective, metaheuristic methods do not guarantee that a global optimum is found, as they may possibly converge early to a local optimum while attempting to solve complex multi-modal problems. Also, their convergence speed and optimality of the solution depends on proper adjustment of parameters of each meta-heuristic method. In order to prevent the issue of getting stuck in local optima, hybrid methods, or variations of a certain method, have been proposed by researchers.

Very significant progress has been done in the research of optimization methods for the ORPD, but further work remains to be done in finding effective and viable methods or improving upon the existing methods, given the great importance of reactive power optimization in power systems.

1.2 – Objectives

The aim of this thesis is to apply a meta-heuristic optimization method for the ORPD, built upon existing research in order to guarantee good results and mostly in an attempt to improve existing meta-heuristic methods for the ORPD. More specifically, the objectives are to:

- Implement a meta-heuristic algorithm which finds the optimal values for the reactive power outputs of shunt capacitors, tap ratios of regulating transformers and voltage of generators in a power system, such that the active power losses are minimized.
- Implement constraint handling methods which guarantee that the optimal solution is feasible, given the importance of constraints in a power system.
- Add improvements to the implemented meta-heuristic algorithm such that the probability of the undesirable early convergence to local optima is decreased, thus increasing the chances of convergence to the global optimum.

1.3 – Framework of the study

The economy of a country is directly related to the amount of electrical energy it generates and consumes. For this reason, it is important to have a power grid which is sufficient, stable and sustainable. There are three main stages in a power grid: generation, transmission and distribution, which deliver electrical power to industrial and residential consumers. A power system needs to remain functional and efficient when facing the dynamic factors within the system. Power losses in a power system usually constitute a very considerable percentage of the generated power, which translates to high costs for the producer. Variations in consumer's power demands cause rapid voltage fluctuations in the system, which compromise the voltage stability in the loads, apparent power flow in the transmission lines and generated reactive power. Therefore, it is necessary to implement means to minimize power losses and to stabilize the voltage. These two processes are realized by regulating the generator voltages, transformer tap settings and capacitive reactive power injection. The purpose of the Optimal Reactive Power Dispatch is to find the optimal values for these system variables.

1.4 – Thesis outline

The research work developed within the scope of this thesis is structured in seven chapters:

Chapter 1 introduces the ORPD, outlining its importance and briefly discussing current solution methods, and the objective of this thesis is presented.

In chapter 2 a survey of several conventional methods, meta-heuristic and hybrid methods commonly employed to solve the ORPD problem is conducted for better understanding of the applicability of each kind of method to the problem, allowing a comparison between these methods in order to select one to implement in this project.

In chapter 3 the mathematical model of the ORPD is defined, as a nonlinear optimization problem, presenting the objective function for active power losses and restrictions, which will be considered the fitness parameter of the metaheuristic algorithm to be implemented.

Chapter 4 explains the concept of particle swarm optimization, presents its mathematical model and its evolution since it was proposed, discusses the advantages and disadvantages of this algorithm, discusses the issue of early convergence to a local optimum in PSO, the selection of parameters and neighborhood topologies, while referencing the work of past researchers about the PSO algorithm. Given the importance of the constraints in power systems, we discuss constraint handling methods for metaheuristic algorithms appropriate for the ORPD.

Chapter 5 describes the PSO algorithm developed in this thesis, explaining the parameter selection, the local version of the PSO designated “Fitness-Distance Ratio PSO” and its mathematical model, which was implemented in an attempt to improve the results of the base PSO, as well as the Second Order PSO implemented for the same reason. Since the ORPD is a mixed-integer optimization problem, a discrete variable updating method is presented and mathematically modeled and a constraint handling method previously mentioned is implemented and explained.

Chapter 6 firstly describes the parameters of the power systems where the developed versions of the PSO were tested for the solution of the ORPD. The results of several runs of each version of the algorithm are displayed and a statistical analysis is conducted in order to compare the performance of each version of the PSO. Finally, an empirical analysis of the results is performed, taking into account what was expected in theory as seen in previous chapters and conclusions are drawn regarding the apparent behavior of the particle swarm deduced from the results.

Chapter 7 summarizes the thesis; the most relevant conclusions are drawn and perspectives of futures work are presented.

Chapter 2 – STATE OF THE ART

2.1 – Numerical methods

Since the introduction of the ORPD, several gradient-based optimization methods have been used to solve this problem, such as interior point methods, nonlinear programming, linear programming and quadratic programming.

Hsu *et al.* [1] proposed a nonlinear programming model of minimization of power losses and capacitor cost. The optimal capacitor sizes are determined with the MINOS software package which uses a linearly constrained Lagrangian method to solve NLP problems.

Noureddine and Chandrasekaran [2] also proposed a nonlinear programming model, where the problem is linearized in terms of capacitor size and voltage changes to eliminate the need to recompute the load flow.

As in the research described above, the gradient-based and Newton methods, mostly interior point methods, are guaranteed to converge to an optimal solution in few iterations and conveniently handle inequality constraints. However, the solution is not guaranteed to be feasible, besides, these methods are sensitive to the initial solution when the objective functions have multiple local minima, and present many limitations when the objective functions and constraints are nonlinear, non-convex and discontinuous [3]. The active power loss minimization problem of the ORPD presents these characteristics. Interior point methods require discrete variables, the capacitor sizes and transformer tap ratios in this case, to be relaxed as continuous in the optimization model. In some methods, the continuous variables obtained from the optimization are rounded off to the nearest integer value by employing a penalty function. Due to the numerical approximation, such approaches may introduce unnecessary increases in the objective functions and even cause constraint violations. In conclusion, the conventional mathematical programming methods are fast and easy to implement but its feasibility cannot be guaranteed and the results may deviate from the global optimal solution, meaning most of them solve nonlinear optimization problems only on an approximate basis [4], [5], [6], [7].

In more recent research, other methods were employed in combination with numerical methods in order to address their issues:

Fan *et al.* [9] implement the interior point filter algorithm combined with complementarity theory to solve the ORPD, where the control variables considered are terminal voltages of generators, outputs of reactive power compensation devices and transformation ratios of on-load tap-changers (OLTC'S). The objective function is of minimization of the active power of the slack bus. The interior point filter algorithm performs a first optimization by treating the discrete control variables as continuous, which are the capacitor size and transformer tap ratio, and then uses complementarity constraints to find the boundaries of the discrete variables. Afterwards, the interior point filter algorithm performs a second optimization taking the previous solution as a starting point. The algorithm was tested on the IEEE 30 bus and IEEE 118 bus systems. The optimized results of voltage and power loss are shown in the following table, as well as calculation times:

Table 1 – Results from "Discrete Reactive Power Optimization Based on Interior Point Filter Algorithm and Complementarity Theory" (base active power is 100 MW)

Test system	**Power loss before optimization calculation (p.u)**	**Power loss after optimization calculation (p.u)**	**Number of buses violated voltage limits after optimization calculation**	**calculation time (s)**
IEEE30	0.0736	0.0706	0	0.75
IEEE118	1.3229	1.0966	0	1.77

Yang *et al.* [10] minimize the active power losses using the predictor-corrector interior point method for global optimization, combined with the branch and bound search algorithm to round the result to the discrete values of capacitor size and transformer tap ratio, as the interior point method is not appropriate for discrete variables. The algorithm was tested in the IEEE 14 bus system, with 6 generators, 4 adjustable transformers and 4 reactive power compensation devices. The optimized power losses are in the following table:

Table 2 – Results from "Reactive Power Optimization of Power System based on Interior Point Method and Branch-bound Method" (base active power is 100 MW)

Loss before optimization (p.u)	0.13396
Loss after optimization (p.u)	0.12271
Computing time (sec)	0.806

The two algorithms developed above proved to be very computationally efficient with a considerable

reduction in active losses, but there is no indication regarding the feasibility of the solution obtained for the test systems.

Combinatorial search methods such as the branch and bound algorithm or the cutting plane algorithm may overcome the limitations of the numerical methods, but they are affected by the "curse of dimensionality", hence are inefficient on power systems with a high number of buses. Metaheuristic optimization methods, as a second-generation of solution techniques for the ORPD, have been proposed with the aim of overcoming the limitations of the gradient-based methods. Among these methods are genetic algorithms, particle swarm optimization, ant colony optimization, simulated annealing and fuzzy logic, among others. These methods are distinct from conventional methods as they have the ability to reach a solution that is as close as possible to the global optimal point, thereby escaping early convergence to local optimal points. Even though the solutions obtained with the objective function calculation are still continuous and then rounded off to a discrete value, the use of appropriate discretization methods and constraint handling techniques guarantee that the (sub) optimal solution is always feasible [8].

2.2 – Metaheuristic methods

Rojas *et al.* [11] propose a genetic algorithm (GA) for the optimal capacitor placement and sizing, where the objective function of minimization of energy losses, considering power loss costs and capacitor costs, is used to determine both the optimal location and size of the capacitor which are encoded in each chromosome.

Al-Hajri *et al.* [12] use a particle swarm algorithm (PSO) to solve the ORPD with objective function of active power loss minimization, with capacitor locations and size as discrete control variables in each particle. The algorithm was tested in a 69 bus distribution system, and the optimal capacitor installation caused an active loss reduction from 225.006 kW to 152.081 kW and also improved the voltage profile.

Kasaei and Gandomkar [13] use an ant colony algorithm (ACO) with an identical objective function used for optimal capacitor placement and size and also feeder configuration.

Zhao *et al.* [14] propose a GA combined with the reduced gradient method (GA-RGM), which was used to improve the local searching capability of the algorithm. The objective function is of

minimization of the active losses and the voltage deviation, where the control variables are the capacitor size to connect to each bus and the transformer tap positions. The algorithm was tested on the IEEE 30 bus system. A comparison between GA, PSO, ACO, and SA (simulated annealing) for this problem is presented in terms of active power loss, out of bus voltages and average calculating time. The method used in this paper has the best results. The second best approach is by ACO and the worst one is by SA. The results are in the following table:

Table 3 – Results from "Reactive Power Optimization by Genetic Algorithm Integrated with Reduced Gradient Method" (base active power is 100 MW)

Method	Active Power Loss (p.u.)	Voltage Out-of-limit	Average Calculating Time(s)
GA	0.07029	1	20.3
ACO	0.07224	0	15.8
SA	0.07319	2	22.5
PSO	0.07285	1	18.6
GA-RGM	0.06869	0	15.5

Bhattaeharya and Goswami [15] use a fuzzy logic system for optimal capacitor placement, selecting capacitor nodes on the basis of line losses for the active power membership function, while for the reactive power membership function the location is identified on the basis of both the reactive power demand and distance of the reactive load from the substation. The capacitor sizes are then determined with SA.

GA explores all possible combinations of solutions, has good capability of finding a global optimum but tends to stay trapped in a local optimum and is slow in its search and is computationally expensive [7] [16].

SA likewise has good global optimum searching capability but tends to escape the local optimum and is slow overall. Therefore, this algorithm is often combined with others to compensate for its issues while making use of its advantages [16] [17].

PSO is easy to implement and has few adjustable parameters, fast convergence, but tends to converge to a local optimum. This is because, in the search process, all particles consider the optimal particle as the goal, then search towards the same direction, which leads to losing the ability to explore unknown areas [17] [18] [19].

ACO has good global optimum searching capability but tends to escape local optima [20], its theoretical analysis is difficult, and although convergence is guaranteed, the convergence time is uncertain.

Fuzzy based approaches use simple logic and are much less computationally intensive then the methods above, but the solution obtained is likely to be sub-optimal, although acceptable depending on the case.

It has been seen that meta-heuristic methods allow one to explore the search space more efficiently, avoiding convergence to local minima and thus ending the search process. However, such methods cannot guarantee that the obtained solution is the global optimum, but often a point very close to the desired point is found. So far there is no method which can determine the global optimal solution for every nonlinear optimization problem. Another disadvantage of methods based on metaheuristics is a need for an excessive number of evaluations of the objective function in the case of optimization of power systems, which generally requires a large amount of exact load flow calculations. In some types of problems, the computational time required can compete with an exhaustive search, in which all possibilities are tested [19].

2.3 – Hybrid methods

Meta-heuristic algorithms can be combined with each other or combined with gradient methods with the goal of compensating each other's shortcomings which were previously described in this section. Combining the characteristics of an algorithm which can efficiently find a local optimum with an algorithm that has good global optimum searching capability, for example, a PSO-SA combination or a gradient-GA combination, one can obtain better optimal solutions than if these algorithms were used separately.

Basyarach *et al* [21] employ loss sensitivity factors which are used to find the best location for the capacitors and a simplified PSO (APSO), which was altered to accelerate convergence, is used to determine the optimal capacitor size, with the following velocity update equation:

$$v_{i,t+1}^{d} = v_{i,t}^{d} + \alpha * rand_t + \beta * \left(p_{g,t}^{d} - x_{i,t}^{d}\right) \quad (1)$$

The algorithm was tested in an 84 bus radial distribution system. The results are presented below:

Table 4 – Results from "Optimal Capacitor Placement and Sizing in Radial Distribution System Using Accelerated Particle Swarm Optimization"

Method	Active Power Loss before optimization (MW)	Active Power Loss after optimization (MW)	Computing Time (s)
APSO (proposed approach)	21.21	15.782	99.74
PSO	21.21	15.89	300.34

Wang *et al* [17] propose a PSO-SA hybrid algorithm to determine to determine both the optimal voltage of generators (continuous variable) and the size of capacitors to place at each bus and the transformer tap ratios (discrete variables), with objective function of active loss minimization with a penalty function for constraint handling. The SA offsets the tendency of the PSO to converge to a local optimum, as the SA will consider solutions worse than the optimal one, while the PSO offsets the slowness of the SA which are due to its restricted conditions for global convergence. The algorithm was tested in the IEEE 14 bus system. The results are in the table below:

Table 5 – Results from "Reactive Power Optimization of Power System based on Improved Particle Swarm Optimization" (base active power is 100 MW)

Control variable	PSOSA	PSO	SA
Gen1 (p.u)	1.1	1.1	1.1
Gen2 (p.u)	1.076	1.078	1.076
Gen3(p.u)	1.049	1.044	1.046
Gen6(p.u)	1.1	1.09	1.063
Gen8(p.u)	1.1	1.1	1.077
Tap4-7(p.u)	0.9875	0.9	1.075
Tap4-9(p.u)	0.9375	1.1	0.9375
Tap5-6(p.u)	0.9625	0.9625	0.975
SC9(p.u)	18	18	18
Active losses (p.u)	0.1231	0.1245	0.1242

The PSO-SA not only reached the best solution compared to the PSO and SA on their own, it also proved to converge in less iterations.

Guo *et al* [22] use a GA-SA hybrid algorithm to determine the discrete variables. The SA algorithm is applied after the mutation operation to increase the speed of local optimum convergence. Then, a soft-

constrained interior point method (IPM) is used to find the continuous variables. The algorithm was tested in the IEEE 118 bus system and compared with the primal-dual interior point method, a GA-SA algorithm GA-IPM hybrid algorithm:

Table 6 – Results from "A Combination Strategy for Reactive Power Optimization Based on Model of Soft Constrain Considered Interior Point Method and Genetic-Simulated Annealing Algorithm" (base power is 100 MW)

	GA-SA	**Primal-dual Interior Point Method**	**GA-IPM**	**Algorithm developed in the paper**
The loss before Optimizing (p.u)	1.3359	1.3359	1.3359	1.3359
The loss after optimizing (p.u)	1.4861	1.1486	1.1530	1.1245
The time of calculation (s)	112.958	12.500	20.125	14.343

Prasanna *et al.* [23] determine the optimal location of capacitors by fuzzy logic, depending on power loss reduction indices and voltage at each bus, and then the capacitor sizes are determined such that the power losses and capacitor costs are minimized, by using a newly proposed second order PSO, where each run of the PSO has its optimal solution optimized by the following run of the PSO.

Wang *et al.* [24] determine the size of capacitors and transformer tap positions with a PSO altered with new parameters in order to improve global convergence, by adding a neigborhood best particle to the velocity update equation:

$$v_{i,t+1}^{d} = \chi * [\omega * v_{i,t}^{d} + c_1 * rand_1 * \left(p_{i,t}^{d} - x_{i,t}^{d}\right) + c_2 * rand_2 * \left(p_{g,t}^{d} - x_{i,t}^{d}\right) + c_3 * rand_3 * \left(p_{n_i,t}^{d} - x_{i,t}^{d}\right)] \quad (2)$$

A GA is then applied to the particles after determining their velocities and positions. The objective function of minimization of active losses also considers investment costs as well as voltage and reactive power variation limits of load buses. The algorithm was tested in the IEEE 14 bus system, but with reactive power compensation devices at buses 1, 3, 6 and 8. The results are in the table below:

Table 7 – “Reactive Power Optimization of Power System based on Improved Particle Swarm Optimization”

Bus	1	2	3	4	5	6	7	8	9
Number of Compensation capacitor	3	-	3	-	-	2	-	3	-
Voltage before optimization (p.u)	0.958	1.019	1.025	1.016	1.028	0.959	0.953	0.965	0.963
Voltage after optimization (p.u)	0.983	1.011	1.013	1.014	1.027	0.981	0.999	0.983	0.983
Network loss before optimization (MW)	18.2099								
Network loss after optimization (MW)	15.1846								

The algorithm was able to reduce active losses and improve the voltage profile.

Filho and Medeiros [19] firstly use a gradient method to determine the size of the capacitors to be installed in a set of nodes, then a GA is used to find the optimal combination of capacitor sizes in this set, where the crossover operation is done with ACO.

If the results obtained in [14], [21] [23] and are observed for example, there can be seen no clear advantage between metaheuristic methods or between numerical methods that makes obvious the choice of one method over the other for the ORPD, although from the research reviewed above it can be observed that hybrid algorithms can obtain better optimal solutions, whether they are a gradient based method combined with a metaheuristic or two metaheuristic methods combined.

2.4 – Conclusion

We can verify that gradient based methods converge much faster than metaheuristics which means they may be preferable when the available time is very short, or when a sub-optimal solution is sufficient. Altough for the metaheuristics, by the conclusions reached in the research mentioned above, several different methods can achieve good solutions on the ORPD, and these methods can be improved by a multitude of means, such as a parameter tuning or combination of algorithms. After considering the state of the art, it can be concluded that at this point in the ORPD research the choice of an algorithm for its solution is only a matter of preference.

Moreover, the “no free lunch” theorem by Wolpert and Macready [31] states that the average performance of any pair of algorithms across all possible optimization problems is identical, meaning that if one algorithm outperforms another on a set of problems, the reverse must be true on the set of all other the problems. In other words, no single deterministic or stochastic algorithm will outperform all other algorithms on every problem.

Chapter 3 – MATHEMATICAL MODELLING OF THE OPTIMAL REACTIVE POWER DISPATCH

3.1 – Objective function and restrictions

The ORPD is generally formulated as single-objective optimization problem to minimize active power losses (transmission losses) in a power system. The control variables considered are the generator (PV bus) voltages (V_G), the ratios of transformer taps (T_k) and shunt capacitor reactive power outputs (Q_c). Then, the objective of this problem is to minimize the active power losses in the system by optimizing these control variables within their limits, such that the load bus voltages (V_i), reactive power of generators (Q_{Gi}) and line apparent power flow (S_{lm}) remain also within their limits in normal operating conditions.

The objective function for active power losses in a power system is formulated as (3):

$$\min\ P_{loss} = \sum_{k=1}^{N_{Br}} G_k[V_i^2 - V_j^2 - 2V_iV_j\cos(\delta_i - \delta_j)] \quad (3)$$

Where k =1, 2,…,N_{Br}, being N_{Br} the total number of branches, P_{loss} is the total active power loss in the transmission lines, G_k is the conductance of branch k between buses i and j, V_i , V_j are the voltage magnitudes at buses i and j and δ_i , δ_j are the load angles at buses i and j.

The ORPD must comply with the power balance of the system as well as its operating limits. The power balance equations ensure that the load demand is met considering the power losses in the system, and are accounted for during the power flow solution. These equations constitute the equality constrains of the problem and are shown as (4) and (5):

$$P_{G_i} - P_{D_i} - V_i\sum_{j=1}^{N} V_j[G_{ij}\cos(\delta_i - \delta_j) + B_{ij}\sin(\delta_i - \delta_j)] = 0 \quad (4)$$

$$Q_{G_i} - Q_{D_i} - V_i \sum_{j=1}^{N} V_j [G_{ij} \sin(\delta_i - \delta_j) - B_{ij} \cos(\delta_i - \delta_j)] = 0 \quad (5)$$

where $j = 1,2,\ldots, N$ being N the total number of buses, P_{Gi} and Q_{Gi} are the real and reactive power of the generators connected to the i-th bus, respectively; P_{Di} and P_{Di} are the real and reactive power of the load connected to the i-th bus, respectively, and G_{ij} and B_{ij} are the branch conductance and susceptance between bus i and bus j, respectively.

The limits of the control variables, generator (PV bus) voltages (V_G), the ratios of transformer taps (T_k) and shunt capacitor reactive power outputs (Q_C), constitute inequality constraints:

$$V_{G_i}^{min} \leq V_{G_i} \leq V_{G_i}^{max} \quad (6)$$

$$T_k^{min} \leq T_k \leq T_k^{max} \quad (7)$$

$$Q_{C_i}^{min} \leq Q_{C_i} \leq Q_{C_i}^{max} \quad (8)$$

The limits of the state variables of the system, which consist of load bus voltages (V_i), reactive power of generators (Q_{Gi}), and line power flow (S_{lm}) are also inequality constraints:

$$V_i^{min} \leq V_i \leq V_i^{max} \quad (9)$$

$$Q_{G_i}^{min} \leq Q_{G_i} \leq Q_{G_i}^{max} \quad (10)$$

$$S_{lm}^{min} \leq S_{lm} \leq S_{lm}^{max} \quad (11)$$

The satisfaction of these constraints is essential for a stable and secure functioning of the power system.

A penalty function can be added to the objective function with purpose of avoiding state variable constraint violations:

$$P_{func} = k_q \sum_{i=1}^{N_G} f(Q_{G_i}) + k_s \sum_{i=1}^{N_{Br}} f(S_{l_m}) + k_v \sum_{i=1}^{N} f(V_i)$$

(12)

where k_q, k_s, k_v are penalty factors adjusted such that the reactive power generation, line flow, and load bus voltages remain within their limits when the optimal solution is obtained.

For each state variable Q_{Gi}, S_{lm}, V_i , the static square penalty function $f(x)$ is determined as (13):

$$f(x) = \begin{cases} 0, & if\ x_{min} \leq x \leq x_{max} \\ (x - x_{max})^2, & if\ x > x_{max} \\ (x_{min} - x)^2, & if\ x < x_{min} \end{cases}$$

(13)

The objective function and constraints are nonlinear. The constraints form a non–convex spatial surface with many local optima, due to this the optimization problem is non–convex, which hinders the ability of classical optimization methods to converge at a global optimum (minimum). This is one of the reasons a metaheuristic method was chosen for the minimization of active power losses.

3.2 – Newton-Raphson method

Solving the power flow is the first requisite for the ORPD. The objective of a power flow study is to determine the voltage magnitude and load angle at each bus in a power system such that the power balance equations are satisfied. Various methods for power flow solution exist such as the Newton-Raphson method, Gauss-Seidel method or the Fast-Decoupled solution method.

The Newton-Raphson method is an iterative method based on Taylor's expansion approximation. For power flow studies the unknown variables are voltage magnitudes and angles at load buses and voltage angles at generator buses, and the scheduled quantities are real and reactive power at generation buses and load buses. The method begins with an initial estimate of the unknown variables. Next, by expanding the power balance equations in Taylor Series and neglecting higher order terms, we get a system of equations expressed as:

$$\begin{bmatrix} \Delta\delta^k \\ \Delta|V_i|^k \end{bmatrix} = J^{-1} \begin{bmatrix} \Delta P_i^k \\ \Delta Q_i^k \end{bmatrix}$$

(14)

Where *J* is the Jacobian matrix of partial derivatives:

$$J = \begin{bmatrix} \frac{\partial P_i}{\partial \delta_i}^k & \frac{\partial P_i}{\partial |V_i|}^k \\ \frac{\partial Q_i}{\partial \delta_i}^k & \frac{\partial Q_i}{\partial |V_i|}^k \end{bmatrix}$$

(15)

ΔP_i and ΔQ_i are the mismatches between calculated and scheduled quantities:

$$\Delta P_i = P_i - V_i \sum_{j=1}^{N} V_j [G_{ij} \cos(\delta_i - \delta_j) + B_{ij} \sin(\delta_i - \delta_j)]$$

(16)

$$\Delta Q_i = Q_i - V_i \sum_{j=1}^{N} V_j [G_{ij} \sin(\delta_i - \delta_j) - B_{ij} \cos(\delta_i - \delta_j)]$$

(17)

Then, the voltage magnitude and angles for the next iteration (k) are calculated as:

$$\begin{bmatrix} \delta^k \\ |V_i|^k \end{bmatrix} = \begin{bmatrix} \Delta\delta^{k-1} \\ \Delta|V_i|^{k-1} \end{bmatrix} + \begin{bmatrix} \Delta\delta^k \\ \Delta|V_i|^k \end{bmatrix}$$

(18)

The iterative process stops when the mismatch between the calculated and scheduled quantities is below a specified tolerance:

$$\begin{bmatrix} \Delta P_i^k \\ \Delta Q_i^k \end{bmatrix} \leq \varepsilon$$

The Newton-Raphson method is preferrable to the other mentioned methods because it's faster and more accurate and requires a smaller number of iterations for convergence, which are independent of the size of the power system, so it is better suited for power flow solutions on large systems. The power flow was solved using Siemens PSSE.

Chapter 4 – PARTICLE SWARM OPTIMIZATION

4.1 – Overview of the particle swarm optimization algorithm

The optimization method selected in this study to solve the optimal reactive power dispatch is based on particle swarm optimization (PSO), due to the extensive research done to this date in its applicability for NLP in several areas of engineering, and also relative ease of implementation and convergence speed [25].

Particle swarm optimization is a stochastic optimization technique based on swarm intelligence which simulates the social behavior of certain groups of animals like birds or fishes. These swarms move in a cooperative way to, for example, find food, and each member of the swarm changes its search pattern according to the learning experiences of its own and other members. This algorithm was introduced by an Electrical Engineer, Russel C. Eberhart, and a Social Psychologist, James Kennedy [26].

The PSO algorithm solves the optimization problem with a population of candidate solutions, designated "particles". Each particle is represented by a point in the Cartesian coordinate system, with a random initial position and velocity. The fitness of each particle is measured by the solution of the objective function obtained with that particle at a certain position. Each particle memorizes its own best position in the problem space (individual best position), as well as the best position obtained so far in the swarm (global best). The optimization process, consists of, at each iteration, changing the position of each particle based on a certain velocity which results from a ponderation between its personal best and global best positions weighted by a cognitive and social factor.

In the continuous space coordinate system, the PSO is mathematically described in the following manner:

Assuming a swarm size of *N*, each particle's position vector in D-dimensional space is $X_i = \left(x_i^1, x_i^2, \dots, x_i^d\right)$, and its velocity vector is $V_i = \left(v_i^1, v_i^2, \dots, v_i^d\right)$. A particle's individual best position is $P_i = \left(p_i^1, p_i^2, \dots, p_i^d\right)$, and the global best position is $P_g = \left(p_g^1, p_g^2, \dots, p_g^d\right)$. In the original version of the algorithm, for a minimization problem, the individual best positions are updated as (17):

$$p_{i,t+1}^{d} = \begin{cases} x_{i,t+1}^{d}, & f(X_{i,t+1}) < f(P_{i,t}) \\ p_{i,t}^{d}, & otherwise \end{cases},$$

(19)

meaning the current position of the particle becomes the new individual best position if it has better fitness than the current individual best position. The same formula is used regarding the global best position.

The position and velocity of each particle is updated with the following equations, in that order:

$$v_{i,t+1}^{d} = v_{i,t}^{d} + c_1 * rand_1 * \left(p_{i,t}^{d} - x_{i,t}^{d}\right) + c_2 * rand_2 * \left(p_{g,t}^{d} - x_{i,t}^{d}\right)$$

(20)

$$x_{i,t+1}^{d} = x_{i,t}^{d} + v_{i,t+1}^{d}$$

(21)

Where *rand* is a random value in the range [0, 1] and c_1 and c_2 are stochastic acceleration terms that attract each particle towards its individual best, *pBest* (p_i) and the global best, *gBest* (p_g), respectively.

The initial version of the PSO algorithm was not very effective in optimization problems, so an improved version of the algorithm was proposed by Shi and Eberhart [27] with the inertia weight (ω) factor introduced to the velocity update formula:

$$v_{i,t+1}^{d} = \omega * v_{i,t}^{d} + c_1 * rand_1 * \left(p_{i,t}^{d} - x_{i,t}^{d}\right) + c_2 * rand_2 * \left(p_{g,t}^{d} - x_{i,t}^{d}\right)$$

(22)

This modification in the algorithm greatly increased its performance. This version of the algorithm is the most extensively used.

A variant was introduced by Clerc and Kennedy [28] with a constriction factor (χ) which ensured the PSO algorithm's convergence and improved the convergence rate:

$$v_{i,t+1}^{d} = \chi * [\omega * v_{i,t}^{d} + c_1 * rand_1 * \left(p_{i,t}^{d} - x_{i,t}^{d}\right) + c_2 * rand_2 * \left(p_{g,t}^{d} - x_{i,t}^{d}\right)]$$

(23)

The velocity update in equation (22) can be analyzed from a sociological perspective: First, it can be seen in the equation that each particle is influenced by its own previous velocity, moving to some degree in an inertial manner depending on the velocity, so the parameter that affects this degree of inertial movement, ω, is called inertia weight. The second component in the sum depends on the distance between the particle's current position and its individual best position. This is called the cognitive component as it represents the movements of the particle according to its own experience, and c_1 is called cognitive acceleration factor. The third component in the sum depends on the distance between the particle's current position and the global best position. This is called the social component as it represents information sharing among the particles in the swarm, having the movement of each particle influenced by the position of another. Therefore c_2 is called social acceleration factor.

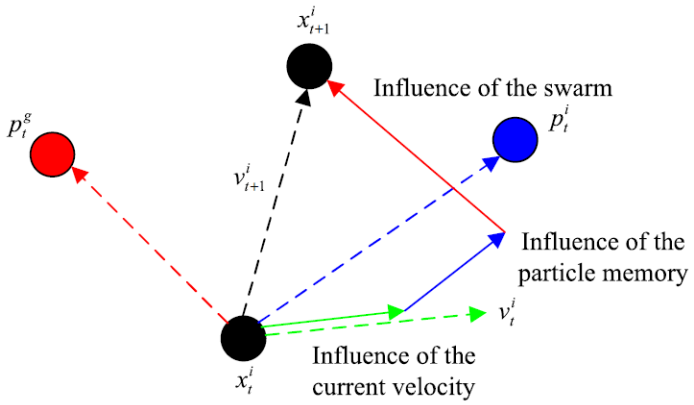

Figure 1 - Illustration of the position and velocity update procedure

The PSO algorithm has two versions, the global version and the local version. The global version is the one formulated initially in (22), where each particle tracks a stochastic average of the *pBest* and the *gBest* particles. In the local version, particles only have information of their individual best and their neighborhood's best. So instead of tracking the global best, they track the neighborhood best, *nBest*, which is the particle with the best fitness in the current particle's neighborhood. This neighborhood is defined by the user.

The global version is faster but might easily converge to a local optimum prematurely. Considering the characteristics of these two versions, one can use global version to get quick results in the first phase of search and then use the local version to refine the search.

Thanks to its intuitive concept, ease of implementation, and wide adaptability to different kinds of objective functions, the PSO algorithm has received a great deal of attention since it was proposed.

Theoretical analysis and application of the algorithm to several domains of engineering has made great progress.

The PSO displays several advantages, which are, summarily: The objective function is not required to be differentiable nor continuous; its convergence speed and rate are high and is easy to implement through programming. Like all the other metaheuristic algorithms, the PSO algorithm has disadvantages: When applied to objective functions with multiple local optima, it has a chance to converge to a local optimum, which is not the best possible solution to the problem. This happens for two reasons: the characteristics of the objective function and, most importantly, disappearance of diversity within the particle population, which causes premature convergence of the algorithm. Studies have shown that each particle oscillates between its individual best position and the global best position until it converges to a point in that interval, generally the global best position. All particles in the swarm behave in the same way, resulting in convergence to a good local optimum of the problem. It is said that the swarm gets "trapped" in a local optimum when the global optimum is not situated between the initial positions and the local optimum that the swarm converges to. In this situation all the particles will be following the local optimum already discovered, resulting in premature convergence. In mathematical terms, a position of a particle becomes equal to both its individual best and the global best for a number of iterations, causing the inertia component in the velocity equation to become zero, doing the same to the velocity equation, and consequently the particle positions no longer change [29]. Therefore, even though the PSO algorithm has the potential for global search, it does not guarantee convergence to the global optimum. Convergence to a global optimum may still occur due to the stochastic nature of the algorithm.

The lack of population diversity was noted by Kennedy and Eberhart [30] as the main factor responsible for the premature convergence to a local optimum, therefore it is important to take measures to increase diversity of the particles so that the algorithm doesn't get trapped in a local optimum. However, enhancing the swarm diversity slows down the convergence because more particles, or more possible solutions, are considered. This phenomenon is well known as it was proven by Wolpert and Macready [31] that there is no single algorithm unarguably better than all the others, for each kind of problem. Therefore, research should not be focused on finding a sort of general mathematical optimizer, but should be instead on finding an optimization algorithm which has a good performance balanced across practical benchmark problems [32].

In order to avoid the premature convergence of the PSO algorithm by increasing the population diversity, while balancing the convergence speed, many variants of the PSO have been proposed. Methods to achieve this include custom tuning of the parameters in the velocity update function,

various formulations of the local version of the PSO and combining the PSO algorithm with other optimization methods or algorithms, as seen in the chapter about the state of the art.

4.1.1 – Parameter selection

It's important to select appropriate values for the parameters in the algorithm, mainly inertia weight ω, constriction factor χ, acceleration factors c_1 and c_2, velocity limits v_{max}, position limits x_{max}, number of particles in the swarm and swarm initialization.

- Inertia weight

Studies have shown that the inertia weight may be the parameter with most impact on the performance of the PSO algorithm. Inertia weight controls the impact of previous historical values of particle velocity on its current one. The inertia weight is used to balance the global and the local search. The larger the inertia weight, the more the algorithm favors global search and the smaller it is, the more it favors local search, so the inertia weight should be set to vary between 0.4 and 1.2 and be gradually reduced as the iterations progress (Shi and Eberhart [27]). A constant inertia weight may not lead to satisfactory results, for this reason researchers have developed various PSO variants with dynamic inertia weights: PSO where the inertia weight decreases linearly along the iterations [27] or with an inertia weight that changes according to a quadratic function (Tang et al. [33]), decreasing according to an exponential function (Lu et al. [34]), according to information from the swarm (Zhan et al. [35]), according to the objective function value (Wang et al. [36]) , an inertia weight that changes with a random factor (Eberhart and Shi [37]), etc.

- Acceleration factors

The acceleration factors c_1 and c_2 represent the weights of the stochastic acceleration terms that pull each particle towards the *pBest* and *gBest*, respectively. Thus, adjustment of these constants changes the amount of "tension" affecting each particle. Low values of the learning factors allow particles to roam far from the target position before being tugged back, while high values result in abrupt movement towards, or away from the target position. In most cases, c_1 and c_2 have the same value, so that the same weight is given to the social and cognitive factors. Following the idea of a varying inertia weight, PSO variants were put forward where the learning factors change with similar methods

(Ratnaweera *et al.* [38]). In some studies, both the acceleration factors and the inertia weight were set to vary simultaneously. In others, the learning factors and inertia weight are determined with optimization techniques such as genetic algorithm (Yu *et al.* [39]), differential evolution (Parsopoulos and Vrahatis [40]), fuzzy logic inference (Juang *et al.* [41]), etc.

- Velocity constraints

The velocity of the particles can be constrained by a maximum velocity v_{max}, which determines the maximum change of the position of a particle at each iteration, therefore it can be used to constrain the global search ability of the swarm. If v_{max} is too high, particles might fly through good solutions. If v_{max} is too small, on the other hand, particles may not explore sufficiently beyond locally good regions. Clerc and Kennedy [42] noted that it was not always necessary to limit the particle velocity, as the constriction factor χ they introduced could realize the same purpose. Some studies showed that better results were obtained by both using the constriction factor and applying velocity limits (Eberhart and Shi [43]). Usually, v_{max} is set to the dynamic range of each control variable and it is a constant value, but it can also linearly decrease along the iterations or decrease according to the success of search history.

- Position constraints

The particles' positions can be constrained by a maximum position, x_{max}, and minimum position, x_{min} in order to stop particles from leaving the feasible region. Robinson and Rahmat-Samii [44]) proposed three control techniques: absorbing wall, reflecting wall, and invisible wall. The absorbing wall sets the velocity in the dimension of a particle to zero if that dimension surpasses the boundaries of the feasible region. The reflecting wall changes the direction of the particle velocity, eventually pulling the particle back to the feasible region. The invisible wall stops the calculation of the fitness of particles that leave the boundaries so that they no longer affect the other particles. The performance of the PSO is greatly influenced by the dimensions of the position vector and the relative position between the global optima and the feasible region boundaries. To solve this problem, some researchers proposed a hybrid boundary by combining the absorbing and the reflecting wall to obtain better performance.

- Population size

The population size is selected according to the problem, even though it does not vary considerably between different problems, meaning a number of particles between 20 and 50 is sufficient for most cases. However, a larger population may be necessary in some cases.

- Initialization

The performance of the PSO algorithm is greatly affected by the initialization of the population. In most problems the initial positions and velocities of the population are randomly generated, but other initialization methods can be employed such as the nonlinear simplex method (Parsopoulos and Vrahatis [45]), orthogonal design (Zhan *et al.* [46]) or centroidal Voronoi tessellations (Richards and Ventura [47]). These methods are used to make the distribution of the initial population as even as possible and to give the algorithm diverse starting points for the algorithm to explore the search space effectively. Robinson et al. [48] noted the PSO and a genetic algorithm could be used one after the other, where the final optimized population of one was the initial population of the other. This proved effective in improving the performance of the algorithm, whether the final algorithm is the PSO or the GA. Yang *et al.* [49] presented a new version of the PSO where the particles were initialized through a low-discrepancy sequence.

4.1.2 – Topology structures

The difference between the global PSO and the local PSO can be explained in terms of their neighborhood topology, and how the neighborhood best position is determined: The global PSO has a star topology, where each particle has the entire swarm as its neighborhood, consequently all particles are attracted to the one global best position; The local PSO has a ring topology, where each particle has its immediate two neighboring particles as its neighborhood, resulting that each particle will be attracted to the best position in its own neighborhood, in this way each particle will follow a different position (usually, as the neighborhoods may overlap). The global PSO and the local PSO are compared for several benchmark objective functions by Engelbrecht [50] in terms of accuracy (best optimal solution reached within a certain number of runs), success rate (percentage of optimal solutions that equaled the specified accuracy), convergence speed and standard deviation, and it was concluded that overall the two versions of the algorithm had similar performance in terms of accuracy, the global PSO

presented slightly better success rate and convergence speed, while the local PSO had better standard deviation. It is concluded that the local PSO is not definitively better than the global PSO for any class of functions, but that it depends on the function in study, and that the neighborhood topology is one of the parameters that have to be tuned.

It has been discussed that the neighborhood topology may improve the performance of the PSO algorithm as it promotes population diversity. So, the design of effective neighborhood topologies is subject of research. The neighborhood can be static or dynamic. A static neighborhood has either a star, ring, or von Neumann topology. A neighborhood is determined according to the index of the particle, or in accordance with the topological distance between the particles. A static neighborhood is used more often by researchers. Kennedy [51] verified that the size of the neighborhood may affect the convergence speed of the algorithm: a PSO with a small neighborhood performs better on problems with a large number of dimensions, while a large neighborhood performs better on problems with a smaller number of dimensions. A version of the local PSO algorithm was proposed by Kennedy [52] with a hybrid space neighborhood and ring topology where the neighborhood best particle was the best among the spatial clustering that it belonged to. Engelbrecht *et al.* [53] compared the *gBest* PSO and the *nBest* PSO in multi-modal optimization problems and found that the *gBest* PSO was incapable of solving this problem while the *nBest* PSO was very inefficient. Mendes *et al.* [54] proposed a PSO where each particle used information of its entire neighborhood to find the optimal solution. Peram *et al.* [55] developed a new variant of the local PSO called fitness-distance ratio PSO, where each dimension of the particles' velocity was updated with an *nBest*, selected to maximize the difference in the fitness of the current particle position and the fitness of the individual best of the neighborhood, divided by the one-dimensional distance between the two positions. This algorithm selected different *nBest* particles in updating each dimension of the velocity vector. Significant research was done about the dynamic topology, several variants were proposed such as PSO with adaptive time varying topology connectivity, where each particle's topology connection changed with time in accordance with the search performance (Lim and Isa [[56]), dynamically adjusted neighborhoods where the size of the neighborhoods increased gradually (Suganthan [57]), dynamically randomized neighborhoods (Lin *et al.* [58]) or a tree hierarchy-based neighborhood (Hanaf *et al.* [59])

4.2 – Constraint handling in metaheuristic algorithms

Metaheuristic algorithm such as the PSO or the genetic algorithm realize an unconstrained search according to their definitions. Therefore, when applied to constrained optimization problems, these algorithms require additional constraint handling methods. If the infeasible individuals were simply discarded along the iterations, a loss of useful information would possibly occur, contributing to the probability of early convergence to local optima. Therefore, by using a constraint handling method, the information of infeasible individuals can be exploited, resulting in better solutions. Several constraint handling methods have been proposed by researchers. Michalewicz and Schoenauer [60] grouped the constraint handling methods for evolutionary algorithms into four categories: preserving feasibility of solutions [61], penalty functions, making a separation between feasible and infeasible solutions, and hybrid methods. The most simple and early constraint handling method in literature is the penalty function approach [62][63]. In this method, a penalty function value is calculated for each individual, this value is then added (in case of minimization) to the objective function so that the individual is penalized if it's infeasible. With a penalty function, the fitness function *F(X)* is usually in the following form (24):

$$F(X) = f(X) + \sum_{j=1}^{m} R_j [v_j(X)]^2 \qquad (24)$$

Where *f(X)* is the objective function, and the penalty function depends on the constraint violation $v(x)$ and the penalty coefficients R_j. The effectiveness of the penalty function depends on the adequate selection of the penalty coefficients, which are generally found by trial and error. The penalty coefficients have to be manually adjusted depending on the problem, and in the case of the ORPD, also depending on the power system. With this approach, selecting the correct penalty coefficients is tedious and prone to error, as small penalty coefficients over-explore the infeasible region, possibly leading to convergence to an infeasible solution, while large penalty coefficients may not explore the infeasible region properly, possibly leading to premature convergence. In order to avoid the issues with the penalty function method, several constraint handling methods with no parameters were proposed, which also showed better results than this method.

4.2.1 – Constraint handling methods

A constrained optimization problem is written in the following form [63]:

$$\min/\max:\ F(X), \qquad X = (x_1, x_2, \dots x_n), X \in S$$

$$s.t: \begin{matrix} g_i(X) \geq 0, & i = 1, 2, \dots p \\ h_j(X) = 0, & j = p+1, \dots m \end{matrix}$$

(25)

Where S is the search space and p and $m - p$ are the number of inequality and equality constraints respectively.

The equality constraints $h_j(X)$ can be converted into inequality constraints, so that the inequality constraints become $G_i(X) = \{g_i(X) \geq 0;\ h_j(X) \geq 0\}$. The overall constraint violation for an individual is a weighted average of all the constraint values [25] expressed as (26):

$$v(X) = \frac{\sum_{i=1}^{m} w_i G_i(X)}{\sum_{i=1}^{m} w_i}$$

(26)

Where w_i is a weight parameter given by $w_i = \frac{1}{G_{max_i}}$ and G_{max_i} is the maximum constraint violation for each constraint:

$$G_{max_i}(X) = \begin{cases} \max\{g_i(X), 0\}, & i = 1, 2, \dots p \\ \max\{|h_j(X)| - \delta, 0\}, & j = p+1, \dots m \end{cases}$$

(27)

Where δ is a tolerance parameter.

Various constraint handling methods that require no manual parameter adjustment have been proposed in literature, such as the principle of constrained non-domination, self-adaptive penalty, ε-constraint, stochastic ranking, or an ensemble of constraint handling techniques.

According to the principle of constrained non-domination seen in [64], candidate solution X_i is said to dominate candidate solution X_j if and only if one of the following scenarios occur:

- X_i is feasible and X_j is infeasible

- Both solutions are infeasible and X_i has a smaller overall constraint violation than X_j
- Both solutions are feasible and X_i is not worse than X_j in objective function value

Therefore, by applying this principle, feasible solutions will always be selected over infeasible solutions. By comparing solutions based on the overall constraint violation, the infeasible solutions will become feasible as the algorithm progresses, necessarily converging to a feasible solution.

In [65] a self-adaptive penalty function is proposed, which requires no parameter tuning. Two types of penalty functions are added to the infeasible individuals in order to identify the infeasible individuals with the least constraint violation. The value of the added penalty depends on the number of feasible individuals currently in the population. The greater the number of infeasible individuals, the greater will be the added penalty.

In [66] the epsilon-constraint method is proposed, where the tolerance of the constraints is controlled with the ε parameter. This parameter is updated until a certain number of generations, where ε will be set to zero in order to only obtain solutions with no constraint violations.

In [67] the stochastic ranking method is proposed which achieves a balance between the objective function value and the overall constraint violation stochastically, through a probability factor that determines whether the objective value or the constraint violation determines the rank of each individual.

4.3 – Conclusions

Much research has been done to this date in improvement methods for the PSO. It has been seen that the PSO is a highly customizable algorithm, and many approaches can be taken in increasing the diversity of the particles, thus decreasing the probability of premature convergence to local optima, and also in constraint handling methods.

Chapter 5 – SOLUTION METHODOLOGY

This chapter describes and justifies the implementation of the PSO algorithm used to solve the ORPD by attempting to minimize the objective function for active power losses described in the third chapter.

5.1 – Problem space coding

The position of a particle *i*, or a possible solution, is a multi-dimensional vector composed of the control variables: reactive power output of shunt capacitor banks, transformer tap ratios, and generator voltages:

$$X_i = \left[\left(Q_{c_1}, Q_{c_2}, \dots Q_{c_{d_c}}\right), \left(T_{k_1}, T_{k_2}, \dots T_{k_{d_t}}\right), \left(V_{g_1}, V_{g_2}, \dots V_{g_{d_g}}\right)\right]$$

(28)

Where d_c is the number of shunt capacitors banks in the system, d_t is the number of transformers with tap regulation, and d_g is the number of generators in the system.
The shunt capacitor reactive power Q_c and transformer tap positions T_k are discrete control variables which can assume one of the values allowed by the step *n* of the capacitor banks and transformer tap positions respectively, forming the vector of discrete values X_{dis_i} for each:

$$X_{dis_i} = [x_{min},\ x_{min} + n,\ x_{min} + 2n, \dots x_{min} + (m-1)n] ,$$

(29)

where x_{min} is the lower limit of the discrete variable X_i and $m = \frac{x_{max} - x_{min}}{n}$.
The state variable vector is composed of the generators reactive power, load bus voltages, and line power flow:

$$Y_j = \left[\left(Q_{G_1}, Q_{G_2}, \dots Q_{G_{d_g}}\right), \left(V_1, V_2, \dots V_{d_b}\right), \left(S_{lm_1}, S_{lm_2}, \dots S_{lm_{d_l}}\right)\right]$$

(30)

Where d_g is the number of generators, d_b is the number of buses and d_l is the number of transmission lines.

5.2 – Parameters

The inertia weight was set to decrease linearly along the iterations, as proposed by Shi and Eberhart [27], with expression (31):

$$\omega = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{iter_{max}} iter$$

(31)

Where ω_{max} = 0.9 and ω_{min} = 0.4, $iter_{max}$ is the maximum number of iterations and *iter* is the number of the current iteration. This means the inertia weight decreases linearly from 0.9 to 0.4. In theory, these values allow a balance between global and local exploration, because with a higher inertia weight, the swarm will realize a global search in the earlier iterations, gradually performing a more local searcher while the iterations progress.

The acceleration factors c_1 and c_2 were both set to 1, meaning the same weight is given to cognitive and social search, in other words, the particles are as compelled to follow the individual best particle as the global best particle. These values were adjusted by experimentation, taking the average optimal solution of several runs into account.

The constriction factor χ proposed by Clerc and Kennedy [27], which usually improves the convergence rate, was applied with expression (32).

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \ ,$$

$$\varphi = \ c_1 + c_2 \qquad (32)$$

The velocity update equation used then, is equation (23):

$$v_{i,t+1}^{d} = \chi * [\omega * v_{i,t}^{d} + c_1 * rand_1 * \left(p_{i,t}^{d} - x_{i,t}^{d}\right) + c_2 * rand_2 * \left(p_{g,t}^{d} - x_{i,t}^{d}\right)]$$

There was no need to apply velocity constraints as the constriction factor used can limit the velocity. Actually, it was found that applying velocity limits with the constriction factor already in place only worsened the results.

The maximum and minimum position, x_{max} and x_{min} were set, for each dimension, to the boundaries of the respective control variable (V_G, T_k, Q_C), according to the inequality constraint equations (6), (7) and (8) . The generator voltages, along with the transformer tap settings, are constrained between 0.9 and 1.1 inclusively, which are the minimum and maximum per unit voltages allowed at each bus by convention.

The number of particles was adjusted for each power system in study by experimentation, starting with a population of 30 particles. For the IEEE 14 bus system, a considerably better average optimal solution was obtained with 60 particles than 30 particles for the same number of runs, although the convergence speed of the algorithm is lower as to be expected. It was found that with 90 particles, the PSO starts to become noticeably slow while the results only improve slightly. Therefore, the number of particles was set to 60. For the IEEE 39 bus system, in line with the previous explanation, the optimal number of particles was found to be 90. The results that led to this decision are in Annex C.

The position and velocity of each particle were initialized randomly in each dimension, within the bounds of the respective control variables. For the capacitor sizes and transformer tap settings, a value was randomly selected from the available discrete values.

It was found, by experimentation, that the initial solution greatly affected the final solution reached by the algorithm.

5.3 – Constraint handling

The equality constraints (4) and (5) (Chapter 3 –) are met by the power flow solution method, in this case the Newton-Raphson method. The inequality constraints for the control variables (6), (7) and (8) (V_G, T_k, Q_c) are met by imposing position limits, after the position update equation is applied to each particle with position X_i, with expression (33):

$$x_{i,t+1}^{d} = \begin{cases} x_{i,t+1}^{d}, & x_{min}^{d} \leq x_{i,t+1}^{d} \leq x_{max}^{d} \\ x_{min}^{d}, & x_{i,t+1}^{d} < x_{min}^{d} \\ x_{max}^{d}, & x_{i,t+1}^{d} > x_{max}^{d} \end{cases} \quad (33)$$

Which means that if any dimension in a new calculated position happens to be above or below the maximum or minimum limit for the respective control variable, that dimension is set to that maximum or minimum limit respectively. In this manner it's ensured that the optimal solution is always feasible, in terms of control variables.

The inequality constraints for the state variables (9), (10) and (11) (V_i, Q_g, S_{lm}) (Chapter 3 –) cannot be handled in the same manner as the control variables, as the state variables are dependent variables. In order to do so, after the power flow is solved and the objective function value (active power loss) is obtained for each particle, the inequality constraints G_i are calculated for each state variable vector Y_j with the following expression:

$$G_i(Y_j) = \begin{cases} 0, & y_{min}^{d} \leq y_j^{d} \leq y_{max}^{d} \\ y_{min}^{d} - y_j^{d}, & y_n^{d} < y_{min}^{d} \\ y_j^{d} - y_{max}^{d}, & y_n^{d} > y_{max}^{d} \end{cases} \quad (34)$$

Then, the overall constraint violation $v(Y_j)$ is calculated for each state variable vector according to equation (26):

$$v(Y_j) = \frac{\sum_{i=1}^{m} w_i G_i(Y_j)}{\sum_{i=1}^{m} w_i}$$

Next, the principle of constrained non-domination (4.2.1 – Constraint handling methods) is applied when the objective value $f(X)$ of current position X_i is compared to the objective value $f(P)$ of the individual best position P_i. For the PSO algorithm, this principle can be translated into the following expression:

$$p_{i,t+1}^{d} = \begin{cases} x_{i,t+1}^{d}, & f(X_{i,t+1}) < f(P_{i,t}) \ \ and \ \ v(Y_{i,t+1}) \leq v\left(P_{Y_{i,t}}\right) \\ x_{i,t+1}^{d}, & v(Y_{i,t+1}) \leq v\left(P_{Y_{i,t}}\right), \\ p_{i,t}^{d}, & otherwise \end{cases}$$

(35)

where P_{Y_i} is the state variable vector obtained with the individual best position P_i. By incorporating the value of overall constraint violation in the expression for the selection of the individual best, using the principle of constrained non-domination, the overall constraint violation functions as an additional objective function which has to be minimized by the state variables. The above expression means that the minimization of the overall constraint violation $v(Y)$ is given priority over the minimization of the actual objective function $f(X)$ (active power losses). This means that the current position will always become the new individual best if it has a smaller overall constraint violation, regardless of the objective function value, in other words a feasible solution is always chosen over an infeasible solution. In this way, over the iterations, the particles that result in any state variable constraint violation will be moved to a position where the overall constraint violation is zero, resulting in guaranteed convergence to a feasible solution, given a sufficient number of iterations. For the ORPD, the importance of applying a constraint handling method that guarantees feasible solutions, such as the principle of constrained non-domination, is that if state variables are outside their feasible region, in other words, either the load bus voltages, generated reactive power, or line flow are outside their limits, the stability and safety of the system will be compromised and lead to failure of the power system.

5.4 – Updating discrete variables

The PSO performs a search on the continuous domain. After updating the position of each particle, the final step of the algorithm is to approximate the control variables Q_c and T_k to their nearest feasible discrete domain locations. The two discrete vectors which constitute the discrete domains are:

$$Q_{c_{dis}} = [Q_1, Q_2, \dots Q_m],$$
$$T_{k_{dis}} = [T_1, T_2, \dots T_m],$$

where m is the number of discrete variables. The method here implemented to update the discrete variables was proposed by Chowdhury, Tong, *et al.* [8]. The location of a particle in the discrete domain is defined by a local hypercube that is expressed as:

$$H_d = \{(x_1^L, x_1^U), (x_2^L, x_2^U), \dots (x_m^L, x_m^U)\},$$

$$x_i^L < x_i < x_i^U, \quad i = 1, 2, \dots m$$

where m is the number of discrete variables, x_i a variable in the current particle position, and x_i^L and x_i^U are two consecutive values in a discrete vector X, which define the vertices of the local hypercube. The total number of vertices in the hypercube is then 2^m.

The Nearest Vertex Approach (NVA) is used to approximate each variable x_i (Q_c and T_k) of the current position to the nearest vertex of its local hypercube H_d based on Euclidean distance, as follows:

$$\tilde{x}_i = \begin{cases} x_i^L, & |x_i - x_i^L| \leq |x_i - x_i^U| \\ x_i^U, & otherwise \end{cases}, i = 1, 2, \dots m \quad (36)$$

With this approximation, the control variables Q_c and T_k in the current solution will become discrete values allowed by the respective discrete vector:

$$\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \dots \tilde{x}_m]$$

In the manner described above, the NVA can also be used to handle binary variables, by having a discrete vector consisting of [0, 1], or in the case of fixed shunt capacitors which are either turned on or off, [0, Q_c].

5.5 – Fitness-distance ratio PSO

It has been seen that the global PSO has the tendency to get trapped in a local optimum, essentially due to all particles following the same global best position. Peram *et al.* [55] addressed this issue by proposing a mixed global-local PSO, called FDR-PSO, where each particle is influenced by

neighboring particles as well as the global best. Although the results obtained with the global PSO[1] were satisfactory, this version of the algorithm was implemented in an attempt to improve the results. In this algorithm, one particle for each velocity dimension is selected to update the velocity of each particle in order to counter the possibility of the movement towards different particles cancelling each other. The selected particles must satisfy two criteria: they must be near the current particle, and they must have visited a position of higher fitness. This particle, for each velocity dimension, is one which maximizes the ratio of the difference between the fitness of the individual best position $f(P_j)$ of the neighborhood and the fitness of the current particle position $f(X_i)$, and the one-dimensional distance between the two positions, hence the name fitness-distance ratio PSO (FDR-PSO).

As such, a particle from the neighborhood with individual best position P_j is selected such that expression (37) is maximized:

$$\frac{f(P_j) - f(X_i)}{\left|P_j^d - X_i^d\right|} \qquad (37)$$

The particle with individual best P_j that maximizes this ratio will become the neighborhood best (*nBest)* particle P_{n_i}, towards which the dimension d of the velocity of particle i is updated. Unlike in the original formulations of the local PSO, the *nBest* does not replace the *gBest* in updating the velocity, instead the influence of the *gBest* is maintained for higher a diversity of particles, therefore a new component is added in the velocity update equation for the *nBest.* The velocity update equation now becomes:

$$v_{i,t+1}^d = \chi * [\omega * v_{i,t}^d + c_1 * rand_1 * (p_{i,t}^d - x_{i,t}^d) + c_2 * rand_2 * (p_{g,t}^d - x_{i,t}^d) + c_3 * rand_3 * (p_{n_i,t}^d - x_{i,t}^d)] \qquad (38)$$

The acceleration factors c_1, c_2 and c_3 were all set to 1, as better results were obtained with equal acceleration factors than, for example, by setting $c_2 = 2$ or $c_3 = 2$ which gives more weight to one of the particles, and the remaining learning factors to 1. By making the acceleration factors equal each particle is equally pulled towards the *pBest*, *gBest* and *nBest* particles.

[1] The original formulation of the PSO will from this point on be designated global PSO

The other parameters of the algorithm are the same as in the global PSO. The FDR-PSO was implemented as an optional function for the global PSO, hence the constraint handling and updating of discrete variables are also maintained.

The neighborhood of each particle consists of a maximum of five particles with a saved individual best position. At each iteration, a new particle is appended to the neighborhood while the oldest particle is removed from the neighborhood such that five particles are present in each neighborhood at all times. The power flow is evaluated for each particle in the neighborhood, meaning the power flow is solved six times for each particle as opposed to only one time in the global PSO. Therefore, the FDR-PSO will be slower than the global best PSO, for this reason the size of the neighborhood must be limited, otherwise the FDR-PSO would take much longer to converge than the global PSO, offsetting its own advantages.

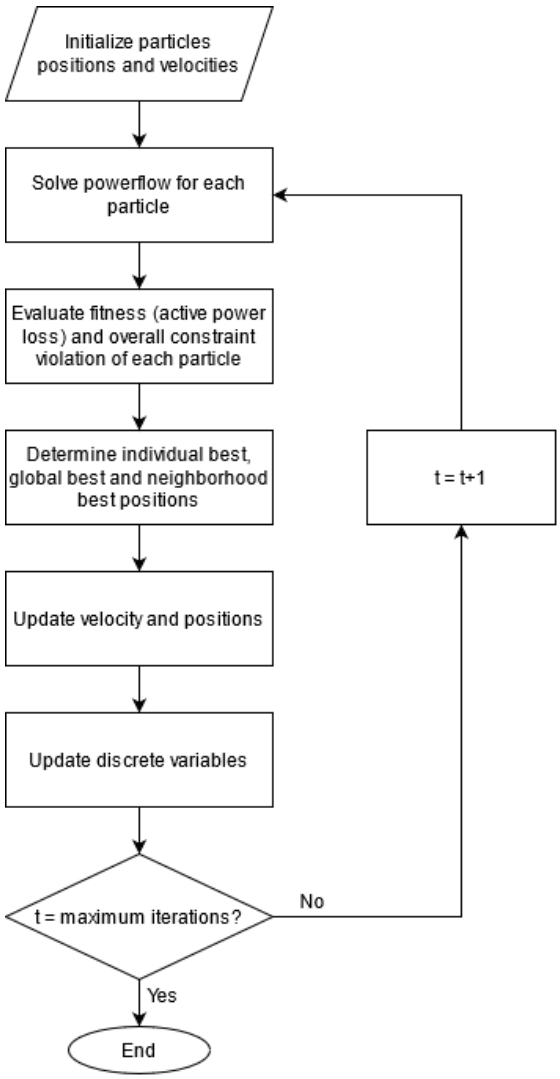

Figure 2 - Flowchart of the implemented global PSO (or FDR-PSO, to be decided by the user)

5.6 – Second order PSO

As seen in Chapter 4 there are numerous methods to improve the performance of the PSO. One method is by improving the initialization of the particles, as explained in 4.1.1 – Parameter selection. The importance of the initialization was noted in literature, particularly by Prasanna *et al.* [23], where the authors proposed a variation of the PSO that optimizes the initialization of the particles, called second order PSO (SO-PSO).

The second order PSO can be described as running the PSO algorithm more than once, where the initial positions for the second run are generated based on the global best position obtained at the end of the first run, maintaining this global best position as the initial global best position. In this manner the algorithm continues the search for an optimum after convergence, having a good solution as a starting point.

The initial position for each particle is generated by making each control variable in the final *gBest* position vary randomly within a certain range, while maintaining the variables within their boundaries. This range is determined for each variable through their minimum and maximum change limit with expressions (39) and (40):

$$minlim_{x_i^d} = \frac{x_{min}^d}{p_{g_{min}}^d} \quad (39)$$

$$maxlim_{x_i^d} = \frac{x_{max}^d}{p_{g_{max}}^d} \quad (40)$$

where x_{min}^d and x_{max}^d are the lower and upper limit of the control variable at dimension *d* and $p_{g_{min}}^d$ and $p_{g_{max}}^d$ are the minimum and maximum value of the control variable at dimension *d* found at the *gBest* position respectively.

Finally, the new initial positions for the next run are calculated with expression (41), for each particle *i*:

$$X_{i_{init}} = p_g^d * rand_i^d \tag{41}$$

$$rand_i^d \in [minlim_{x_i^d}, maxlim_{x_i^d}]$$

The stopping criterion of the implemented SO-PSO is a maximum number of runs of the global PSO. The maximum number of runs was set to 10. By experimentation, this stopping criterion was found sufficient in order to obtain a good optimal solution at convergence in the final run.

All the parameters in the velocity update equation were maintained, as the SO-PSO is also implemented as an optional function for the global PSO. In this manner the SO-PSO can be combined with the FDR-PSO.

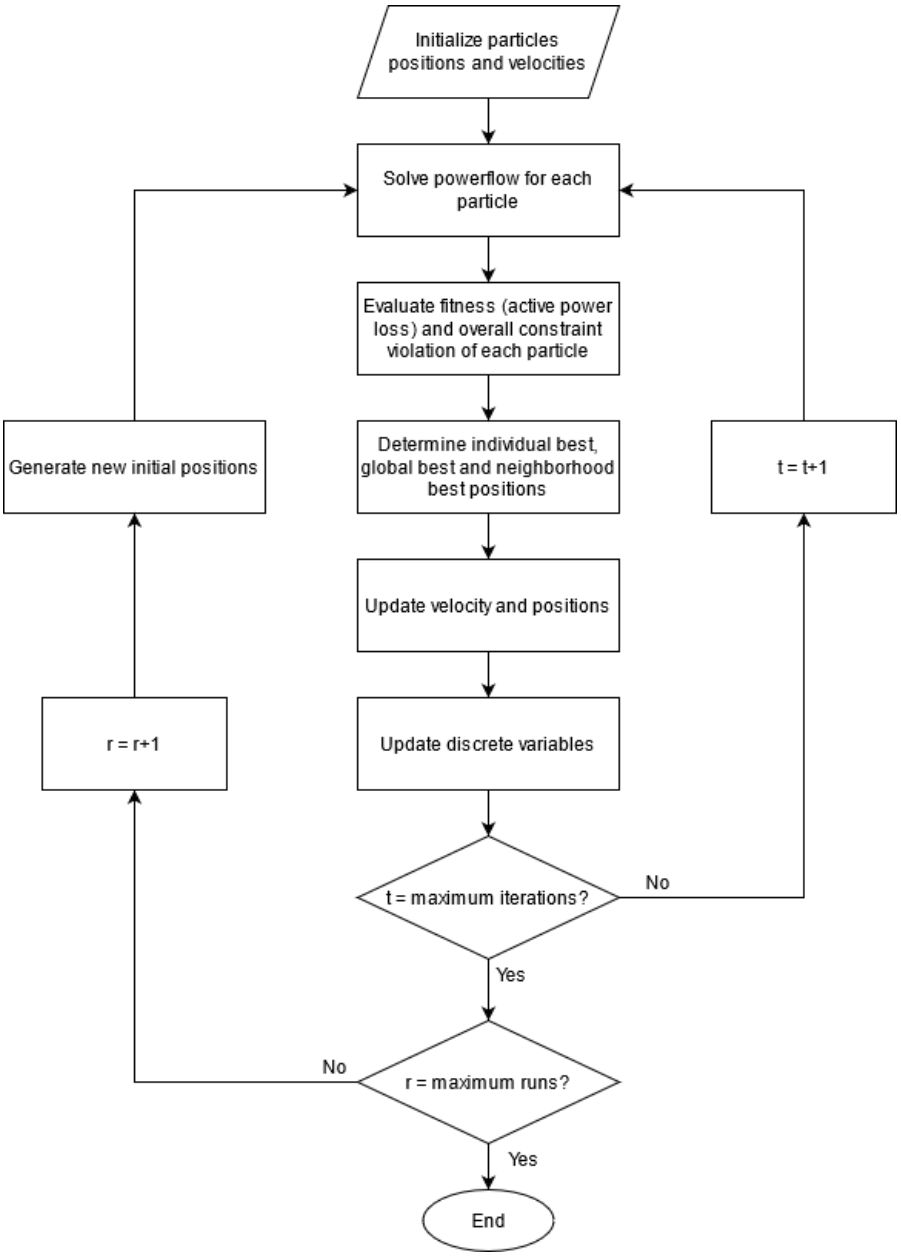

Figure 3 - Flowchart of the implemented SO-PSO (or FDR-SO-PSO, to be decided by the user)

5.7 – Software implementation

The power systems for testing of the algorithm were implemented in Siemens PSS E Xplore 34. The algorithm was written in Python 2.7, and the code interacts with PSS E through its API, which allows the changing of the control variables in the system, solving the power flow, reading the state variables and calculating the power losses. PSSE Xplore only allows power systems of up to 50 buses.

The PSO algorithm was created with an object-oriented paradigm: each particle is an instance of the "Particle" class which includes attributes such as the current position, velocity, individual best position, current fitness, best fitness, neighborhood best position, etc. The methods in the "Particle" class execute the operations of the PSO algorithm previously discussed: objective function evaluation, velocity update, position update, discrete variable updating, etc. The FDR-PSO, SO-PSO and the FDR-SO-PSO are made possible through optional methods, such that the FDR and SO aspects are functionalities of the PSO which can be toggled on or off. See Annex F for the Python code.

Chapter 6 – CASE STUDIES

6.1 – Test systems data and parameters

The versions of the PSO implemented as explained in the previous chapter were tested on the IEEE 14 bus and IEEE 39 bus systems.

For the IEEE 14 bus system, a maximum of 50 iterations of the PSO were sufficient for convergence. The number of particles was set to 60, for reasons explained chapter 5.1 –PSO. The SO-PSO has a maximum of 10 iterations, totaling 500 iterations for the PSO.

The IEEE 14 bus system has: five generators, at buses 1, 2, 3, 6 and 8; three transformers with tap regulation between buses 4-7, 4-9 and 5-6; one shunt capacitor at bus 9. The full network data is in Annex B.1. The voltage limits on all buses are [0.9, 1.1] p.u; the tap step of the regulating transformer is 0.0125 p.u and the tap position limits are [0.9, 1.1] p.u; the shunt capacitor battery has three equal groups, each with 6 Mvar. In accordance with these control variables, each particle has 9 dimensions, forming a vector with the following structure:

$$X_i = \left[\left(Q_{c_1}\right), \left(T_{k_1}, T_{k_2}, T_{k_3}\right), \left(V_{g_1}, V_{g_2}, V_{g_3}, V_{g_4}, V_{g_5}\right)\right]$$

And the discrete vectors for the shunt capacitor sizes, $Q_{c_{dis}}$, and tap ratios, $T_{k_{dis}}$, will be:

$$Q_{c_{dis}} = [0, 6, 12, 18]\ Mvar$$

$T_{k_{dis}} =$
$[0.9, 0.9125, 0.925, 0.9375, 0.95, 0.9625, 0.975, 0.9875, 1, 1.0125, 1.025, 1.0375, 1.05, 1.0625, 1.075, 1.0875, 1.1]\ p.u$

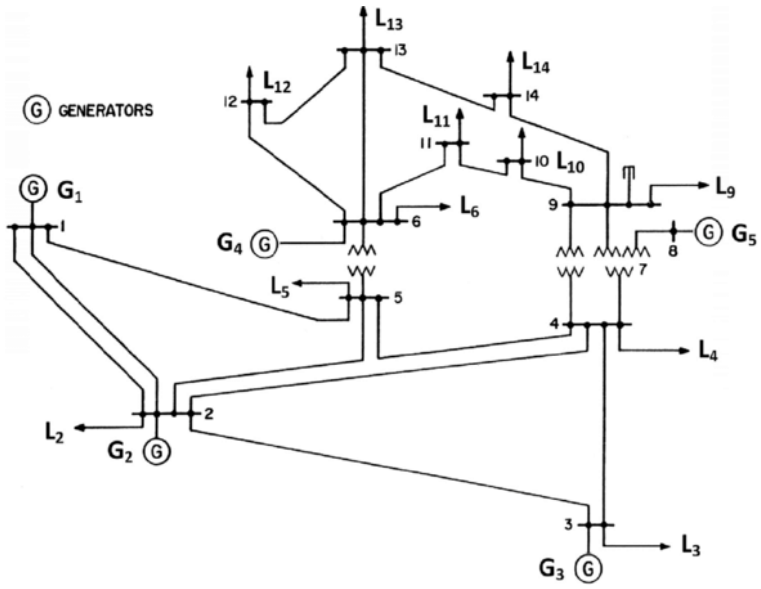

Figure 4 - Single line diagram for IEEE 14 bus system

For the IEEE 39 bus system, a maximum of 100 iterations of the algorithm were sufficient for convergence. The number of particles was set to 90. The SO-PSO has a maximum of 10 iterations, totaling 1000 iterations for the PSO.

The IEEE 39 bus system has: ten generators, at buses 30, 31, 32, 33, 34, 35, 36, 37, 38, 39; one transformer with tap regulation between buses 6-31; two shunt capacitors at buses 4 and 5. The full network data is in Annex B.2. The voltage limits on all buses are [0.9, 1.1] p.u; the step of the transformer tap positions and the tap position limits are the same as in the IEEE 14 bus system; The shunt capacitor battery at bus 4 has four equal groups, each with 25 Mvar, and at bus 5 it has eight groups, each with 25 Mvar. In accordance with these control variables, each particle has 13 dimensions, forming a vector with following structure:

$$X_i = \left[\left(Q_{c_1}, Q_{c_2}\right), \left(T_{k_1}\right), \left(V_{g_1}, V_{g_2}, V_{g_3}, V_{g_4}, V_{g_5}, V_{g_6}, V_{g_7}, V_{g_8}, V_{g_9}, V_{g_{10}}\right)\right]$$

And the discrete vectors for the shunt capacitor sizes, $Q_{c_{dis}}$, and tap ratios, $T_{k_{dis}}$, will be:

$$Q_{c_{dis_1}} = [0, 25, 50, 75, 100]\ Mvar$$

$$Q_{c_{dis_2}} = [0, 25, 50, 75, 100, 125, 150, 175, 200]\ Mvar$$

$T_{k_{dis}} =$
$[0.9, 0.9125, 0.925, 0.9375, 0.95, 0.9625, 0.975, 0.9875, 1, 1.0125, 1.025, 1.0375, 1.05, 1.0625, 1.075, 1.0875, 1.1]\ p.u$

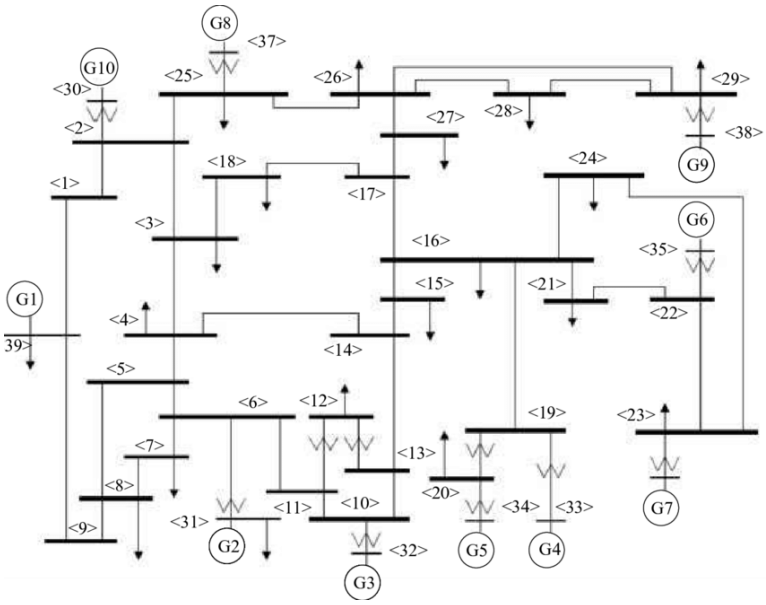

Figure 5 - Single line diagram for IEEE 39 bus system

6.2 – Results and analysis

6.2.1 – Performance analysis

A statistics-based analysis is required in order to evaluate the performance of a stochastic algorithm such as the PSO. The PSO, FDR-PSO, SO-PSO and FDR-SO-PSO were ran 30 times for the IEEE 14 bus system and IEEE 39 bus system. The optimal solutions at the end of each run and corresponding objective function values were registered. The performance of the algorithms was measured with the following indicators:

- Average accuracy: the average objective function value (active power losses) of all the runs, obtained at convergence
- Standard deviation (consistency) in relation to the average accuracy of the 30 runs
- Best accuracy: the best objective function value (lowest active power losses) obtained within 30 runs of the algorithm

- Success rate: percentage of the total runs where the best accuracy was reached. Two decimal places in the objective function value were considered in calculating the success rate.
- Convergence time: the approximate average time elapsed in each run

The results of 30 runs with all versions of the algorithm are displayed in Table 8 and Table 9, with the power losses in MW, along with the performance indicators mentioned above. For the statistical analysis, 60 particles were used for the IEEE 14 bus system and 90 particles for the IEEE 39 bus system for reasons explained in the previous chapter. The results obtained with higher numbers of particles are in Annex C. The results obtained considering that the available capacitor battery can only be turned on or off (discrete vector is [0, X_{max}]) are seen in Annex E.

Table 8 - Optimized power losses for IEEE 14 bus

Optimized power losses [MW]				
Run no.	**PSO**	**FDR-PSO**	**SO-PSO**	**SO-FDR-PSO**
1	12,339	12,287	12,280	12,280
2	12,475	12,324	12,307	12,280
3	12,339	12,292	12,307	12,280
4	12,534	12,305	12,320	12,280
5	12,329	12,297	12,307	12,280
6	12,510	12,295	12,307	12,280
7	12,430	12,334	12,280	12,281
8	12,392	12,293	12,280	12,280
9	12,280	12,296	12,307	12,280
10	12,437	12,291	12,280	12,280
11	12,398	12,291	12,307	12,280
12	12,372	12,281	12,280	12,280
13	12,320	12,281	12,280	12,280
14	12,473	12,308	12,280	12,280
15	12,471	12,291	12,281	12,280
16	12,339	12,300	12,280	12,280
17	12,318	12,339	12,280	12,280
18	12,392	12,289	12,307	12,280
19	12,280	12,284	12,280	12,280
20	12,297	12,285	12,280	12,280
21	12,395	12,302	12,280	12,280
22	12,437	12,312	12,320	12,280
23	12,348	12,300	12,396	12,280
24	12,392	12,300	12,280	12,280
25	12,420	12,299	12,280	12,280
26	12,360	12,282	12,392	12,280

27	12,398	12,281	12,280	12,280
28	12,325	12,307	12,288	12,280
29	12,373	12,292	12,307	12,280
30	12,437	12,286	12,280	12,280
Performance indicators				
	PSO	**FDR-PSO**	**SO-PSO**	**SO-FDR-PSO**
Average accuracy [MW]	12,387	12,297	12,298	12,280
Std. dev.	0,066	0,015	0,030	0,000
Best accuracy [MW]	12,280	12,281	12,280	12,280
Success %	7	17	57	100
Time [sec]	28	140	280	1400

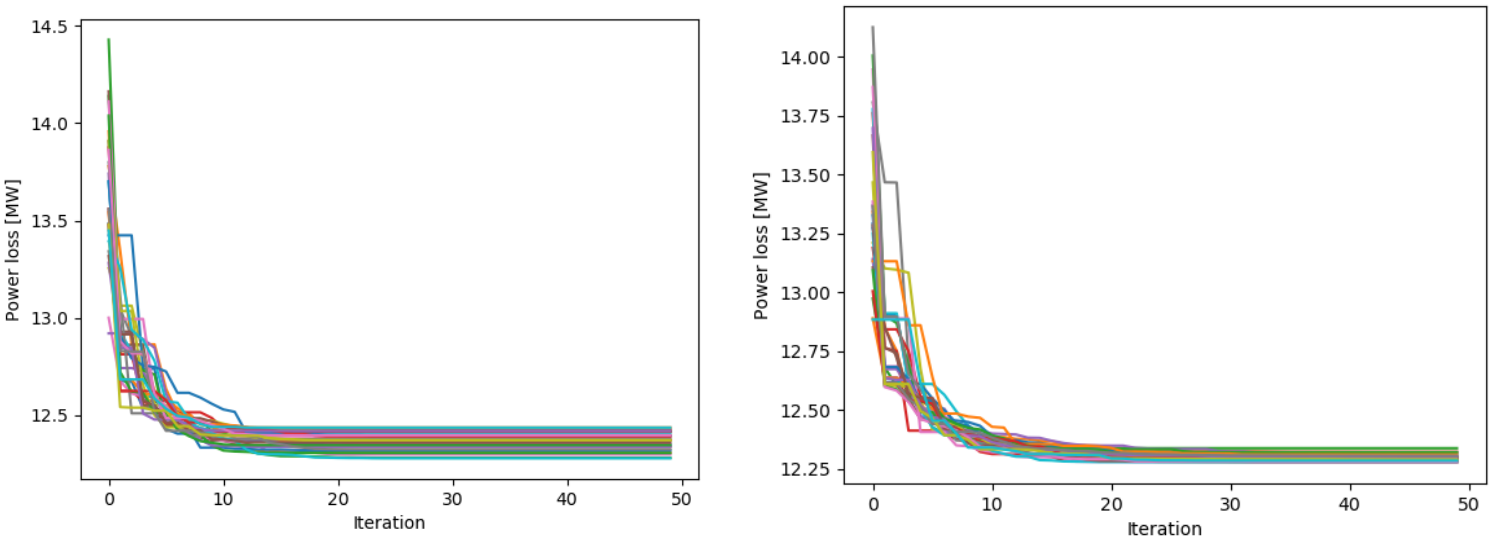

Figure 6 - Convergence characteristics of 30 runs of the global PSO (left) against that of the FDR-PSO (right) in the IEEE 14 bus system

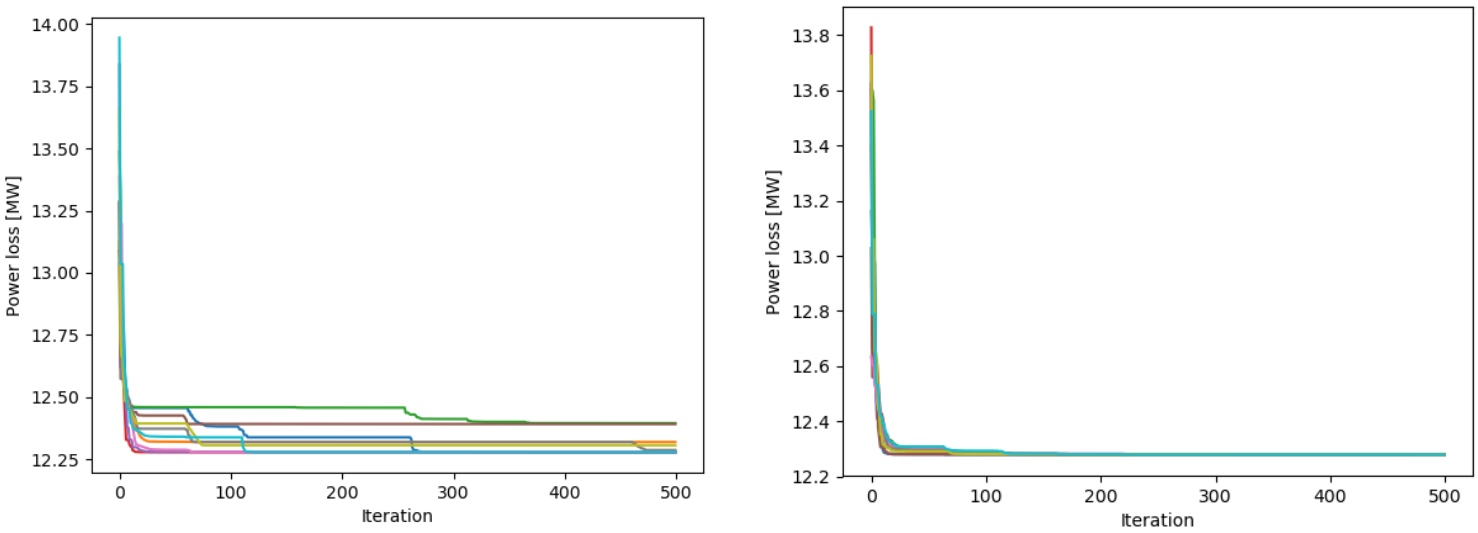

Figure 7 - Convergence characteristics of 10 runs of the SO-PSO (left) against that of the SO-FDR-PSO (right) in the IEEE 14 bus system.

Table 9 - Optimized power losses for IEEE 39

Optimized power losses [MW]				
Run no.	**PSO**	**FDR-PSO**	**SO-PSO**	**SO-FDR-PSO**
1	38,063	38,151	38,019	37,949
2	38,042	38,153	37,966	37,949
3	38,215	38,081	38,013	37,952
4	38,022	38,138	37,951	37,955
5	38,259	38,054	37,969	37,954
6	38,030	38,093	37,981	37,950
7	37,991	38,167	37,981	37,949
8	38,082	38,043	38,015	37,949
9	38,047	38,116	37,987	37,950
10	38,555	38,501	38,018	37,966
11	38,015	38,177	37,975	37,954
12	38,000	38,086	37,957	37,963
13	37,962	38,096	37,948	37,955
14	38,242	38,069	38,043	37,951
15	38,013	38,035	37,983	37,951

16	38,111	38,042	38,057	37,976
17	38,074	38,153	38,007	37,951
18	38,036	38,163	38,015	37,949
19	37,990	38,149	37,983	37,949
20	38,184	38,076	38,090	37,964
21	38,071	38,051	38,022	37,956
22	38,085	38,039	37,989	37,953
23	37,991	38,246	38,059	37,966
24	38,125	37,975	37,987	37,949
25	38,004	38,212	38,019	37,957
26	38,017	38,026	38,036	37,950
27	38,648	38,062	38,055	37,950
28	38,352	38,046	38,058	37,950
29	38,734	38,062	38,000	37,963
30	38,059	38,213	37,984	37,955
Performance indicators				
	PSO	**FDR-PSO**	**SO-PSO**	**SO-FDR-PSO**
Average accuracy [MW]	38,134	38,116	38,006	37,955
Std. dev.	0,197	0,097	0,036	0,007
Best accuracy [MW]	37,962	37,975	37,948	37,949
Success %	3	3	7	63
Time [sec]	93	700	930	7000

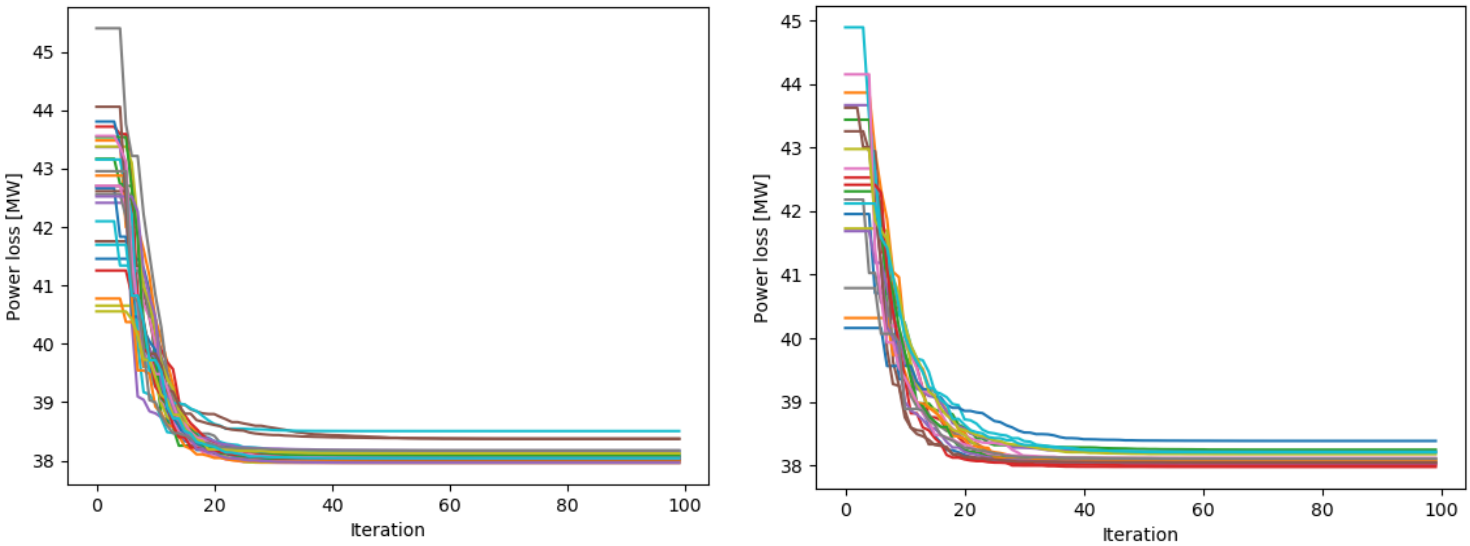

Figure 8 - Convergence characteristics of 30 runs of the global PSO (left) against that of the FDR-PSO (right) in the IEEE 39 bus system

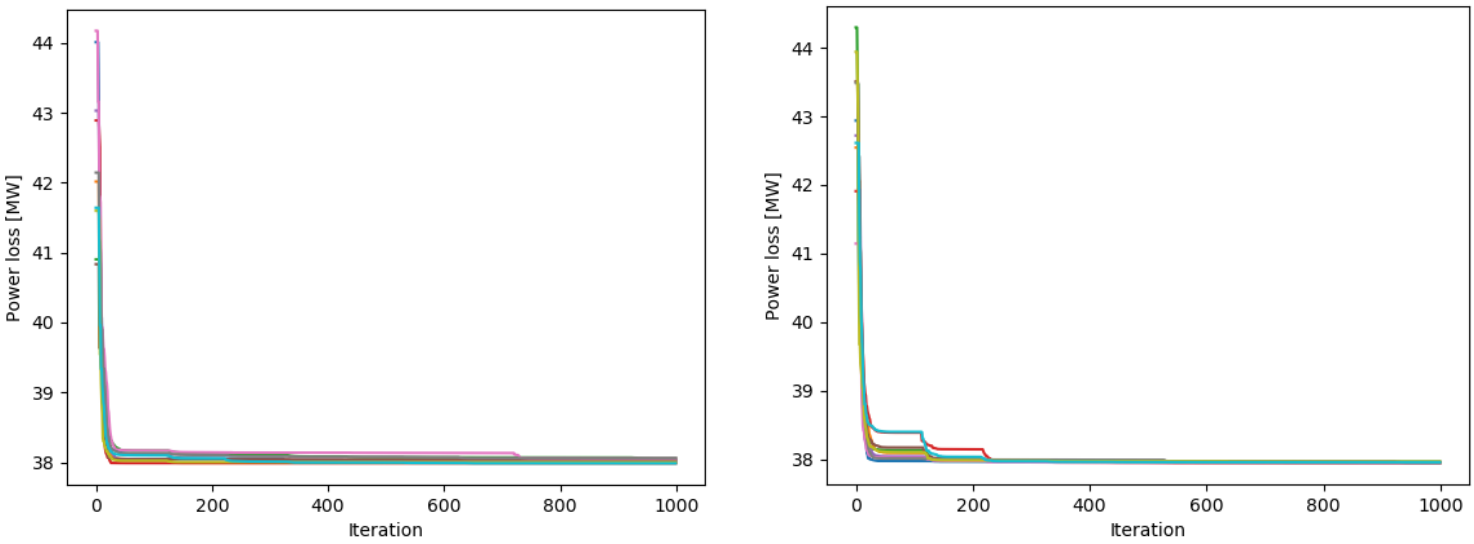

Figure 9 - Convergence characteristics of 10 runs of the SO-PSO (left) against that of the SO-FDR-PSO (right) in the IEEE 39 bus system.

The optimal solutions and state variables on each run are in Annex A.1 and A.2 respectively because they are not relevant for the performance analysis and to not impact reading fluidity.

The performance indicators mentioned above are used to compare the PSO, the FDR-PSO, the SO-PSO and the SO-FDR-PSO.

For the 14 bus system, the FDR-PSO presents clearly better average accuracy and better consistency than the PSO. The best accuracy is practically equal between the two. The success rate of the FDR-PSO is slightly higher. On the other hand, the FDR-PSO is much slower than the PSO, as anticipated.

The SO-PSO has slightly better average accuracy and a practically equal best accuracy and consistency in relation to the FDR-PSO. The success rate of the SO-PSO is much higher than in the FDR-PSO.

The SO-FDR-PSO has slightly better average accuracy than the SO-PSO and equal best accuracy. The standard deviation is much lower and the success rate is 100%.

For the 39 bus system, the FDR-PSO has slightly better average accuracy and considerably better consistency than the global PSO, while the PSO has slightly better average accuracy.

The SO-PSO presents much better average accuracy and a better best accuracy and consistency than the other versions of the PSO, although the success rate is lower than in the FDR-PSO.

The SO-FDR-PSO presents, once again, better average accuracy and consistency and practically equal best accuracy in relation to the SO-PSO, while the success rate is much higher than in the other versions.

6.2.2 – Empirical analysis

Foremost, it can be verified that all the control variables (Annex A.1 – Optimal Solutions) are within their respective bounds, also the discrete control variables, Q_c and T_k assumed one of the values from their discrete vectors, therefore all the obtained solutions are feasible both in the continuous domain and the discrete domain. All the state variables (Annex A.2 – State Variables) obtained with each solution are also in their feasible domain, which means that the constraint handling methods employed (position limits, principle of constrained non-domination) and the nearest vertex approach, as explained in sub-chapters 5.3 – Constraint Handling and 5.4 – Updating Discrete Variables, are effective in this problem. Table 3 displays an example of an optimal solution.

Table 10 - Optimal solutions in the 1st run on the IEEE 14 bus, in [[Mvar], [p.u], [p.u]]

Run no.	PSO	FDR-PSO	SO-PSO	FDR-SO-PSO
1	[[18], [1.0125, 0.9, 0.9875], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[18], [1.0, 0.95, 0.9875], [1.1, 1.086, 1.057, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]

By observing the results displayed above and the performance indicators, an explanation can be made regarding the solutions obtained and the behavior of the swarm in each version of the algorithm:

Firstly, it can be noted in Table 8 that the global PSO occasionally converged to a solution that resulted in losses greater than 12.45 MW, while the best accuracy was 12.28 MW in all versions of the algorithm. Since the initial losses are in the 12.8-14.2 MW range approximately (Figure 6), it appears that convergence to, for example, 12.48 MW is a case of premature convergence to a local optimum, which is one of the issues of the global PSO which had been anticipated. This behavior can be confirmed by observing Figure 10, where we can notice that the global PSO converged around the 10th iteration, to ~12.58 MW, while the FDR-PSO only converged at around the 25th iteration to ~12.30 MW (note that this behavior is occasional for the global PSO, as seen in Table 8, therefore not representative of an average run). Therefore, it can be inferred that convergence to a solution that results in losses close to or greater than 12.50 MW are cases of premature convergence to local optima.

The FDR-PSO appears to have solved the premature convergence issue as originally intended, as there isn't a single power loss close to 12.50 MW within the results, so it appears the FDR-PSO never converges prematurely to local optima as a result of the added influence of the *nBest* particle in updating the velocity for each particle. Statistically, this translates to better consistency and average accuracy, which can also be visualized in Figure 8. We can conclude the FDR-PSO succeeded in improving the performance of the PSO as seen by these indicators.

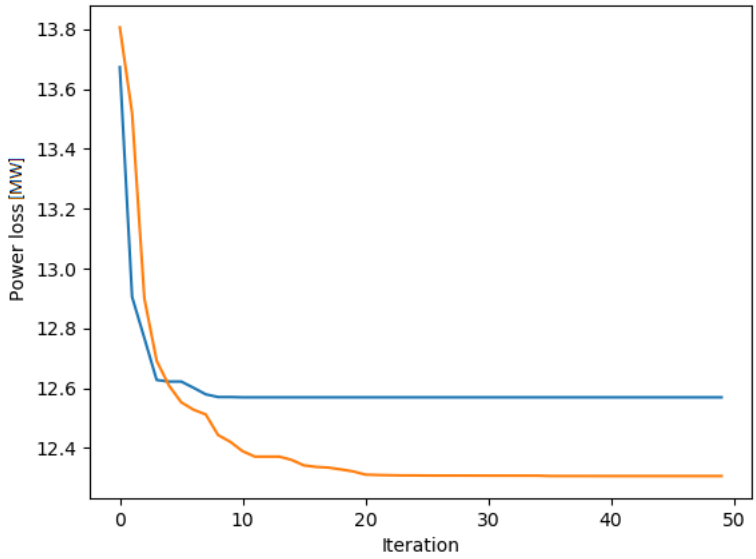

Figure 10 - Convergence characteristics of global PSO (blue) against FDR-PSO (orange) in the IEEE 14 bus system, exemplifying a situation of premature convergence to a local optimum.

The SO-PSO stands out in its success rate, in other words, it converged to the best optimal solution found very often. By continuing the search for an optimum after the PSO converges, with new initial positions generated from the *gBest* at convergence, as explained in the previous chapter, the SO-PSO consistently found a better solution than on the previous run of the PSO, as seen in Annex D. This proves the importance of the initialization of the positions in the PSO algorithm, and most importantly it appears to solve every issue in the original PSO formulation as there appears to be no chance of premature convergence to a local optimum. The SO-FDR-PSO further enhances these results, with the best average accuracy, consistency (practically no standard deviation) and a success rate of 100%, meaning the SO-FDR-PSO consistently reached the best solution obtained so far, which results in losses of 12.28 MW. By observing these results, the SO-FDR-PSO, until this point, appears to be a deterministic algorithm in practice.

Through considerate observation of the results in Table 8 we can safely assume that the entire search space was explored, to the best of the capability of the PSO. All of the optimal solutions are between exactly 12,27964 MW and ~12,7 MW. The additional results in Annex C support this statement, as the best optimal solution obtained was still exactly 12,27964 MW with higher numbers of particles. The solutions obtained with the SO-FDR-PSO confirm this. As explained in 5.6 – Second order PSO, at the end of each run (convergence) of the PSO, new initial positions are generated from the global best position such that these are close to the global best position, and from these new positions the particle swarm will continue the search for a better *gBest*, or optimal solution, in the next run. If a better *gBest* is found, the optimization process will continue as the particles will now have a new position to follow. This mechanism of the SO-PSO can be better understood by Table 11. This table shows that within 10 runs of the PSO a new *gBest* is always found eventually. We can see that through this mechanism, the swarm cannot find an optimal solution that results in losses lower than exactly 12,27964 MW, leading us to conclude that the searching capability of this PSO was indeed exhausted, and that this optimal solution is possibly the global optimum of the problem, or at least very close to it.

Table 11 - Detailed results of one run of the SO-PSO and SO-FDR-PSO for the IEEE 14 bus system. See Annex D for the results of 10 runs of the SO-PSO. Four decimal places are displayed to verify if the algorithms reached the lowest power losses they could find.

Optimized power losses [MW]			
SO-PSO run no.	**Run no.**	**PSO**	**FDR-PSO**
21	**1**	12,45717	12,28151
	2	12,38260	12,27964
	3	12,33905	12,27964
	4	12,33905	12,27964
	5	12,33905	12,27964
	6	12,28078	12,27964
	7	12,27964	12,27964
	8	12,27964	12,27964
	9	12,27964	12,27964
	10	12,27964	12,27964

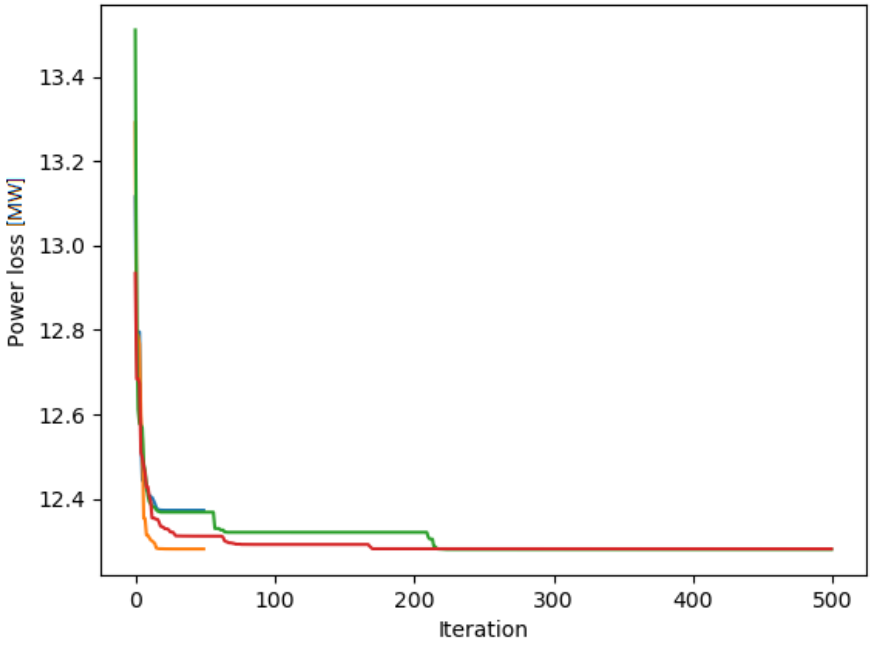

Figure 11 - Convergence characteristics of the global PSO (blue) against FDR-PSO (orange), SO-PSO (green) and SO-FDR-PSO (red) in the IEEE 14 bus system. The curve of the global PSO coincided with that of the SO-PSO.

In the case of the 39 bus system (Table 9), considering that the initial power losses are in the 40-45 MW range Figure 8 and the best accuracy is 37.96 MW, we can still evidence cases of premature convergence to local optima in the global PSO, such as 38.65 MW. The FDR-PSO remains clearly more consistent than the global PSO, although the other performance indicators are identical, meaning that we cannot affirm that the FDR-PSO will certainly not converge prematurely to local optima, but we can affirm that it has a smaller probability to do so. The overall advantage of the SO-FDR-PSO stands as observed by the clearly superior performance indicators: the average accuracy is still the best as well as the consistency and success rate, although the latter is not 100% as seen in the 14 bus system, therefore we cannot conclude that the SO-FDR-PSO is deterministic, although the consistency and success rate are very high. Even though in the 39 bus system the superiority of the FDR-PSO over the global PSO was not as clear, unlike in the 14 bus system, the clear superiority of the SO-FDR-PSO over the SO-PSO can still be observed. Upon comparison of the results between these two versions of the algorithm it can still be observed that the SO-FDR-PSO converges to the best accuracy more consistently than the SO-PSO, hence the better average, consistency and success rate.

By making the same observations that were made for the 14 bus system regarding the search space and the best optimal solution, for the 39 bus system the global optimum should be close to approximately 37.948 MW, as no better solution could be reached with the SO-PSO. In this the case best optimal solution could not be identified as accurately possibly due to the higher dimensionality of the problem.

Table 12 - Detailed results of one run of the SO-PSO and SO-FDR-PSO for the IEEE 39 bus. See Annex D for the results of 10 runs of the SO-PSO. Four decimal places are displayed to verify if the algorithms reached the lowest power losses they could find.

Optimized power losses [MW]			
SO-PSO run no.	**Run no.**	**PSO**	**FDR-PSO**
24	**1**	37,98787	38,38863
	2	37,98776	38,14513
	3	37,98776	37,98034
	4	37,98764	37,95499
	5	37,98720	37,95298
	6	37,98695	37,94978
	7	37,98693	37,94970
	8	37,98693	37,94968
	9	37,98693	37,94912
	10	37,98654	37,94888

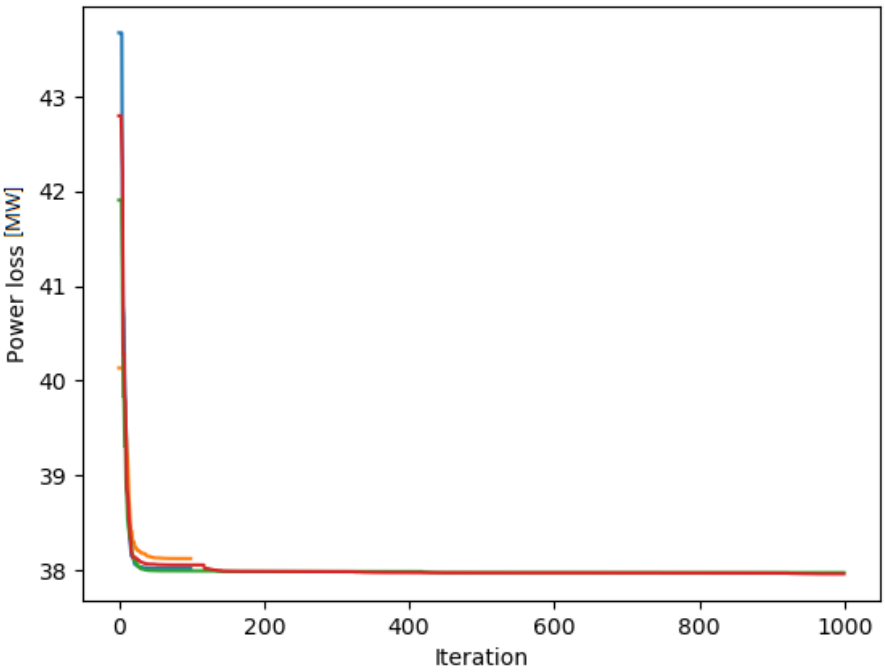

Figure 12 - Convergence characteristics of the global PSO (blue) against FDR-PSO (orange), SO-PSO (green) and FDR-SO-PSO (red) in the IEEE 39 bus system

6.3 – Conclusions

The analysis above allows us to conclude that, for the ORPD with minimization of active power losses as the objective function, the FDR-PSO is less likely to converge prematurely to local optima and consequently its performance will be superior, and more interestingly, that the SO-FDR-PSO is guaranteed to converge to a solution close to the global optimum, as it proved to be capable of exploring most of the search space.

It is worth considering the convergence time of the algorithms in comparing their performance. In solving the ORPD the number of evaluations of the objective function has the greatest impact on the convergence time of the PSO, as each requires a power flow solution, which is a complex operation. For this reason, as expected, the original PSO is the fastest one as it requires the least number of power flow solutions and the slowest one is the FDR-SO-PSO. A greater size of the power system, which corresponds to a greater dimensionality of the problem, will impact the convergence time, as seen by the increase of the convergence times in the 39 bus system in relation to the 14 bus system.

The preferable version of the PSO algorithm for the ORPD as measured by the performance indicators is, in every aspect, the SO-FDR-PSO, if we don't consider the convergence time. In a practical setting, the convergence time may have to be considered and certain parameters will have to be adjusted such as neighborhood size, number of iterations of the SO-PSO or a different stopping criterion, so that a compromise is reached between convergence time and quality of the optimal solutions. In short, the convergence times registered are not to be taken as constants as they are heavily dependant on the parameters used and mostly on the processing power and memory of the system used to run the algorithm (the algorithms were tested on a personal computer with outdated hardware, an Intel Core i5-2450M CPU @2.50 GHz and 8 GB of RAM). Taking this into consideration, the global PSO is suitable for operation of the tested power systems, seeing that its convergence time is less than 2 minutes and can be shortened as explained.

It was not possible to test the algorithm on IEEE systems with more buses because of the 50 bus limit on PSSE Xplore.

It can be noticed that the performance of all the versions of the PSO degraded noticeably from to 14 bus system to the 39 bus system (Table 9), nonetheless in the 39 bus system the algorithm was still able to explore most of the search space and thus converge to solutions close to the global optimum. Therefore it can be said that the complexity of the problem increases linearly with the number of buses, or control variables, of the power system, which allows us to infer that the implemented PSO will converge to, at least, a good local optimum, on a system with any number of buses.

Chapter 7 – CONCLUSION

This thesis was focused on the research and implementation of a metaheuristic algorithm for the solution of the Optimal Reactive Power Dispatch (ORPD), which is a constrained mixed-integer non-linear optimization problem, with minimization of active power losses as the objective function. A particle swarm optimization algorithm (PSO) was implemented to find an optimal solution to the problem. A constraint handling method and a discrete variable updating method were integrated within the PSO algorithm to ensure that the solution is feasible. The developed PSO for the solution of the ORPD was tested on the IEEE 14 bus and IEEE 39 bus systems. The algorithm proved to be effective as it always converged to optimal solutions, which were always feasible both in the continuous domain and the discrete domain, as the control variables and state variables were within their bounds. A version of the local PSO, called Fitness-Distance Ratio PSO (FDR-PSO) was implemented in order to improve the solutions. Another version of the PSO, called Second Order PSO (SO-PSO) was implemented in order to improve the solutions further. A comparison of the results obtained with these versions of the PSO was made through a statistics-based analysis, and it was found that the FDR-PSO obtained improved solutions as expected. The FDR-PSO consistently avoided premature convergence to local optima, solving the primary issue of the PSO, allowing us to conclude that the FDR-PSO is superior to the PSO when applied to the ORPD. The SO-PSO, and in particular the SO-FDR-PSO presented the most noteworthy results. This version of the algorithm consistently converged to what appears to be the global optimum of the problem. The very good accuracy and precision of the SO-FDR-PSO is not commonly seen in metaheuristic algorithms in solving the ORPD, such that the SO-FDR-PSO is close to being deterministic in the solution of this problem. Even though the long convergence time puts the viability of this version of the PSO in question for the ORPD, it is possible that by adjusting certain parameters the convergence time may be decreased while maintaining good accuracy and precision and perhaps improving it further. This meta-optimization is beyond the scope of this project, however, the potential of the SO-PSO has been outlined.

Although it was only possible to test the PSO and its other versions on two systems, it proved to be stable in both systems and found optimal solutions, so we can safely assume that the developed algorithm is scalable, meaning it should find a feasible optimal solution in a power system of any size.

Future work will focus on solving the ORPD with other objective functions such as voltage profile improvement, voltage stability improvement or system costs minimization. Attention will be given in

particular to the SO-PSO since it has proven itself to be promising. Improvements remain to be made in the SO-FDR-PSO as mentioned earlier in order to improve its viability. Another possibility is applying the principle of the second order optimization to a different metaheuristic algorithm for the solution of the ORPD.

BIBLIOGRAPHY

[1] Hsu, C. T.; Yan, Y. H.; Chen, C. S.; Her, S. L.; "Optimal Reactive Power Planning for Distribution Systems with Nonlinear Loads", Proceedings of the 10th IEEE Region Conference on Computer, Communication, Control and Power Engineering, pp. 330-333, 1993;

[2] A.H Noureddine; A. Chandrasekaran, "Linear Programming Approach to Loss Minimization and Capacitor Sizing Placement", 1992;

[3] Gondzio, Jacek, "Interior point methods 25 years later", European Journal of Operational Research, pp. 587-601, 2012,

[4] Lu, Fang; An, Luo; Xu, Xianyong; Fang, Houhui; "Neural Network and Nonlinear Prime Dual Interior Algorithm for Optimization Reactive Power Compensation", Proceedings - International Conference on Electrical and Control Engineering, ICECE 2010, pp. 3898-3901, 2010

[5] Fan, Zheng; Chang, Xiqiang; Wang, Heng; Pu, Tianjiao; Yu, Ting; Liu, Guangyi; "Discrete Reactive Power Optimization Based on Interior Point Filter Algorithm and Complementarity Theory", POWERCON 2014 - 2014 International Conference on Power System Technology: Towards Green, Efficient and Smart Power System, Proceedings, pp. 210-214, 2014

[6] Yang, Zhifang; Bose, Anjan; Zhong, Haiwang; Zhang, Ning; Xia, Qing; Kang, Chongqing; "Optimal Reactive Power Dispatch with Accurately Modeled Discrete Control Devices: A Successive Linear Approximation Approach", IEEE Transactions on Power Systems, pp. 2435-2444, 2017

[7] Cui, Ting; Sun, Yuanzhang; Xu, Jian; Huang, Lei; “Reactive Power Optimization in Power System Based on Improved Niche Genetic Algorithm”, Zhongguo Dianji Gongcheng Xuebao/Proceedings of the Chinese Society of Electrical Engineering, pp. 43-50, 2011

[8] Chowdhury, S.; Tong, W.; Messac, A.; Zhang, J.; “A mixed-discrete Particle Swarm Optimization algorithm with explicit diversity preservation”, in: Structural and Multidisciplinary Optimization, 2013, pp. 367-388, 20

[9] Fan, Zheng; Chang, Xiqiang; Wang, Heng; Pu, Tianjiao; Yu, Ting; Liu, Guangyi; “Discrete reactive power optimization based on interior point filter algorithm and complementarity theory”, POWERCON 2014 - 2014 International Conference on Power System Technology: Towards Green, Efficient and Smart Power System, Proceedings, pp. 210-214, 2014

[10] Yang, Huping; Gu, Yin; Zhang, Yang; Bi, Zhipeng; “Reactive power optimization of power system based on interior point method and branch-bound method”, PEITS 2009 - 2009 2nd Conference on Power Electronics and Intelligent Transportation System, 2009

[11] L. Rojas, R. Garcia, L. Roa, “Optimal Capacitor Location for Radial Systems using Genetic Algorithms”, 2006 IEEE PES Transmission and Distribution Conference and Exposition: Latin America, TDC'06, pp. 1-4, 2006;

[12] AlHajri, M. F.; AlRashidi, M. R.; El-Hawary, M. E.; “A Novel Discrete Particle Swarm Optimization Algorithm for Optimal Capacitor Placement and Sizing”, Canadian Conference on Electrical and Computer Engineering, pp. 1286-1289, 2007;

[13] Kasaei, M. J.; Gandomkar, M.; “Loss Reduction in Distribution Network Using Simultaneous Capacitor Placement and Reconfiguration With Ant Colony Algorithm”, Asia-Pacific Power and Energy Engineering Conference, APPEEC, pp. 10-13, 2010;

[14] Zhao, Dongmei; Pei, Wang; Xu, Zhang; “Reactive Power Optimization by Genetic Algorithm Integrated with Reduced Gradient Method”, Proceedings - 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications, WARTIA 2014, pp. 838-841, 2014;

[15] Bhattaeharya, S. K.; Goswami, S. K. ; “Improved fuzzy based capacitor placement method for radial distribution system”, 2008 Joint International Conference on Power System Technology POWERCON and IEEE Power India Conference, POWERCON 2008, pp. 1-5, 2008

[16] Liu, Yutian; Ma, Li; Zhang, Jianjun; “GA/SA/TS Hybrid Algorithms for Reactive Power Optimization”, 2000

[17] Wang, Zhenshu; Li, Linchuan; Li, Bo; “Based on Particle Swarm Optimization and Simulated Annealing Combined Algorithm for Reactive Power Optimization”, Asia-Pacific Power and Energy Engineering Conference, APPEEC, 2009

[18] Kim, Taegyun; Lee, Yunhwan; Lee, Byongjun; Song, Hwachang; Kim, Taekyun; “Optimal Capacitor Placement considering Voltage Stability Margin based on improved PSO Algorithm”, 2009 15th International Conference on Intelligent System Applications to Power Systems, ISAP '09, pp. 1-5, 2009

[19] Filho, M. C. Pimentel; Medeiros, M. F.; “A Memetic Algorithm Based on Mixed Ant Colony Optimization and Genetic Algorithm for Optimal Capacitor Placement”, 2011 16th International Conference on Intelligent System Applications to Power Systems, ISAP 2011, 2011

[20] Filho, M. C. Pimentel; De Lacerda, E. G.M.; Medeiros, M. F. ; “Capacitor placement using Ant Colony Optimization and gradient”, 2009 15th International Conference on Intelligent System Applications to Power Systems, ISAP '09, pp. 1-4, 2009

[21] Basyarach, Niken Adriaty; Penangsang, Ontoseno; Soeprijanto, Adi; “Optimal Capacitor Placement and Sizing in Radial Distribution System Using Accelerated Particle Swarm Optimization”, 2017 International Seminar on Intelligent Technology and Its Application: Strengthening the Link Between University Research and Industry to Support ASEAN Energy Sector, ISITIA 2017 – Proceeding, pp. 93-97, 2017

[22] Guo, Liya; Ding, Xiaoqun; Chen, Guangyu; “A Combination Strategy for Reactive Power Optimization Based on Model of Soft Constrain Considered Interior Point Method and Genetic-Simulated Annealing Algorithm”, Proceedings - 2010 International Conference of Information Science and Management Engineering, ISME 2010, pp. 151-154, 2010

[23] Prasanna, M. Y. Balaji; Balaji, T. R.Nitish; Kannan, S. M.; “Maximization of Annual Savings by using Fuzzy-Second Order PSO based Optimal Capacitor Placement in RDF for Constant Load”, 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science: Innovation for Humanity, SCEECS 2012, 2012

[24] Wang, Hao; Jiang, Huilan; Xu, Ke; Li, Guodong. “Reactive Power Optimization of Power System based on Improved Particle Swarm Optimization”, DRPT 2011 - 2011 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, pp. 606-609, 2011

[25] Dong, Ying; Tang, Jiafu; Xu, Baodong; Wang, Dingwei. “An Application of Swarm Optimization to Nonlinear Programming”, Computers and Mathematics with Applications, pp. 1655-1668, 2005

[26] Eberhart RC, Kennedy J, “A new optimizer using particle swarm theory.” In: Proceedings of the 6th international symposium on micromachine and human science, pp 39–43, Nagoya, Japan, Mar 13–16, 1995

[27] Shi Y, Eberhart RC (1998), "A modified particle swarm optimizer." In: Proceedings of the IEEE international conference on evolutionary computation, pp 69–73, Anchorage, Alaska, USA,May 4–9, 1998

[28] Clerc M; Kennedy J (2002) "The particle swarm-explosion, stability and convergence in multi dimensional complex space." IEEE Trans Evolut Comput 6(2):58–73

[29] F. van den Bergh and A. Engelbrecht, "A New Locally Convergent Particle Swarm Optimizer," in Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 2002, pp. 96–101.

[30] Kennedy J, Eberhart RC (1995); "Particle swarm optimization?", Proceedings of the IEEE international conference on neural networks, pp. 1942–1948, Perth, Australia

[31] Wolpert, D. H; Macready, W.G; "No Free Lunch Theorems for Optimization" IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, April 1997, pp. 67-82

[32] Garcia-Martinez C, Rodriguez FJ (2012); "Arbitrary function optimization with metaheuristics: no free lunch and real-world problems." Soft Comput 16:2115–2133

[33] Tang Y,Wang Z, Fang J (2011) "Feedback learning particle swarm optimization." Appl Soft Comput 11:4713–4725

[34] Lu J, Hu H, Bai Y (2015a) "Generalized radial basis function neural network based on an improved dynamic particle swarm optimization and adaboost algorithm." Neurocomputing 152:305–315

[35] Zhan Z, Zhang J, Li Y, Chung HH (2009) "Adaptive particle swarm optimization." IEEE Trans Syst Man Cybernet Part B Cybernet 39(6):1362–1381

[36] Wang Q,Wang Z,Wang S (2005) “A modified particle swarm optimizer using dynamic inertia weight.” China Mech Eng 16(11):945–948

[37] Eberhart RC, Shi Y (2001) “Particle swarm optimization: developments, applications and resources.” In: Proceedings of the IEEE congress on evolutionary computation (CEC 2001), pp 81–86, Seoul,Korea, May 27–30

[38] Ratnaweera A, Halgamuge S, Watson H (2004) “Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients.” IEEE Trans Evolut Comput 8(3):240–255

[39] Yu H, Zhang L, Chen D, Song X, Hu S (2005) “Estimation of model parameters using composite particle swarm optimization.” J Chem Eng Chin Univ 19(5):675–680

[40] Parsopoulos KE, Vrahatis MN (2002) “Recent approaches to global optimization problems through particle swarm optimization.” Nat Comput 1:235–306

[41] Juang YT, Tung SL, Chiu HC (2011) “Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions.” Inf Sci 181:4539–4549

[42] Clerc M, Kennedy J (2002) “The particle swarm-explosion, stability and convergence in a multi dimensional complex space.” IEEE Trans Evolut Comput 6(2):58–73

[43] Eberhart RC, Shi Y (2000) “Comparing inertia weights and constriction factors in particle swarm optimization.” In: Proceedings of the IEEE congress on evolutionary computation (CEC 2000), pp 84–88, San Diego, CA, USA, July 16–19, 2000

[44] Robinson J, Rahmat-Samii Y (2004) “Particle swarm optimization in electromagnetics.” IEEE Trans Antennas Propag 52(2):397–407

[45] Parsopoulos KE, Vrahatis MN (2002) “Initializing the particle swarm optimizer using the nonlinear simplex method.” WSEAS Press, Rome

[46] Zhan Z, Zhang J, LiY, ShiY(2011) “Orthogonal learning particle swarm optimization.“ IEEE Trans Evolut Comput 15(6):832–847

[47] Richards M, Ventura D (2004) “Choosing a starting configuration for particle swarm optimization.” In: Proceedings of 2004 IEEE international joint conference on neural networks, pp 2309–2312, Budapest, Hungary, July 25–29, 2004

[48] Robinson J, Sinton S, Rahmat-Samii Y (2002) “Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna.” In: Proceedings of 2002 IEEE international symposium on antennas propagation, pp 31–317, San Antonio, Texas, USA, June 16–21, 2002

[49] Yang C, Gao W, Liu N, Song C (2015) “Low-discrepancy sequence initialized particle swarm optimization algorithm with high-order nonlinear time-varying inertia weight.” Appl Soft Comput 29:386–394

[50] Engelbrecht, A. P. (2013) “Particle swarm optimization: Global best or local best?” Proceedings - 1st BRICS Countries Congress on Computational Intelligence, BRICS-CCI 2013, pp. 124-135

[51] Kennedy J (1999) “Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance.” In: Proceedings of the IEEE international conference on evolutionary computation, pp. 1931–1938, San Diego, CA, Mar 10–13

[52] Kennedy J (2000)” Stereotyping: Improving particle swarm performance with cluster analysis.” In: Proceedings of the IEEE international conference on evolutionary computation, pp. 303–308

[53] Engelbrecht AP, Masiye BS, Pampard G (2005) Niching ability of basic particle swarm optimization algorithms. In: Proceedings of 2005 IEEE Swarm Intelligence Symposium (SIS 2005), pp 397–400, Pasadena, CA, USA, June 8–10, 2005

[54] Mendes R, Kennedy J, Neves J (2004) "The fully informed particle swarm: simpler maybe better." IEEE Trans Evolut Comput 8(3):204–210

[55] Peram T, Veeramachaneni k, Mohan CK (2003) "Fitness-distance-ratio based particle swarm optimization." In: Proceedings of 2003 IEEE swarm intelligence symposium, pp 174–181, Indianapolis, Indiana, USA, April 24–26, 2003

[56] Lim W, Isa NAM (2014) "Particle swarm optimization with adaptive time-varying topology connectivity." Appl Soft Comput 24:623–642

[57] Suganthan PN (1999) "Particle swarm optimizer with neighborhood operator." In: Proceedings of the Congress on Evolutionary Computation, pp 1958–1962, Washington, D.C. USA, July 6–9, 1999

[58] Lin Q, Li J, Du Z, Chen J, Ming Z (2006a) A novel multi-objective particle swarm optimization with multiple search strategies. Eur J Oper Res 247:732–744

[59] Hanaf I, Cabrerab FM, Dimanea F, Manzanaresb JT (2016) "Application of particle swarm optimization for optimizing the process parameters in turning of peek cf30 composites." Procedia Technol 22:195–202

[60] Z. Michalewicz; M. Schoenauer; "Evolutionary algorithms for constrained parameter optimization problems", Evolutionary Computation 4 (1996) 1–32.

[61] S. Koziel, Z. Michalewicz, “Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization” Evolutionary Computation 7 (1999) 19–44

[62] Vishnu, M; Sunil Kumar, T. K.; “An improved solution for reactive power dispatch problem using diversity-enhanced particle swarm optimization”, 2015 IEEE International Conference on Advancements in Power and Energy (TAP Energy), Kollam, India, 24–26 June 2015; pp. 170–175.

[63] K. Deb, “An efficient constraint handling method for genetic algorithms”, Computer Methods in Applied Mechanics and Engineering 186 (2000), 311–338

[64] Deb K, Pratap A, Agarwal S, Meyarivan T (2002) “A fast and elitist multi-objective genetic algorithm”: Nsga-II. IEEE Trans Evol Comput 6(2):182–197

[65] R. Farmani, J.A. Wright, “Self-adaptive fitness formulation for constrained optimization”, IEEE Transactions on Evolutionary Computation 7 (2003) 445–455.

[66] T. Takahama, S. Sakai, “Constrained optimization by the constrained differential evolution with gradient-based mutation and feasible elites”, Presented at the IEEE Congress on Evolutionary Computation, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 2006.

[67] T.P. Runarsson, X. Yao, “Stochastic ranking for constrained evolutionary optimization”, IEEE Transactions on Evolutionary Computation 4 (2000) 284–294

ANNEXES

Annex A – Optimal solutions and state variables for the case studies

The optimal solutions were rounded for better readability. For this reason the obtained power losses with the solutions displayed here may differ from the results in Table 8 and Table 9 in the decimal places.

A.1 – Optimal Solutions (control variables)

In Table 13 are displayed the optimal solutions (control variables) in each run, as [[Q_{c1}], [T_{k1}, $T_{k2,}$ T_{k3}], [V_{g1}, V_{g2}, V_{g3}, $V_{g4,}$ V_{g5}]] in [[Mvar], [p.u], [p.u]] for the 14 bus system:

Table 13 - Optimal solutions for IEEE 14 bus, in [[Mvar], [p.u], [p.u]]

Run no.	PSO	FDR-PSO	SO-PSO	FDR-SO-PSO
1	[[18], [1.0125, 0.9, 0.9875], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[18], [1.0, 0.95, 0.9875], [1.1, 1.086, 1.057, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
2	[[6], [1.0875, 1.0125, 1.1], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[18], [1.075, 0.95, 1.025], [1.1, 1.087, 1.06, 1.072, 1.1]]	[[18], [1.1, 0.9, 1.0], [1.1, 1.087, 1.058, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.086, 1.056, 1.1, 1.1]]
3	[[18], [1.0125, 0.9, 0.9875], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[18], [1.025, 0.925, 1.0], [1.1, 1.086, 1.058, 1.086, 1.1]]	[[18], [1.1, 0.9, 1.0], [1.1, 1.087, 1.058, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
4	[[18], [0.925, 0.9625, 0.975], [1.1, 1.1, 1.1, 1.1, 0.972]]	[[18], [0.975, 1.0375, 1.0], [1.1, 1.086, 1.057, 1.079, 1.1]]	[[18], [0.9625, 1.1, 1.0125], [1.1, 1.086, 1.056, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
5	[[12], [1.1, 0.9, 1.0125], [1.1, 1.087, 1.058, 1.1, 1.092]]	[[18], [1.075, 0.9, 1.0], [1.1, 1.087, 1.058, 1.08, 1.091]]	[[18], [1.1, 0.9, 1.0], [1.1, 1.087, 1.058, 1.087, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
6	[[0], [1.05, 1.0375, 1.1], [1.1, 1.1, 1.056, 1.1, 1.073]]	[[18], [1.0125, 0.9625, 1.0], [1.1, 1.087, 1.057, 1.08, 1.1]]	[[18], [1.1, 0.9, 1.0], [1.1, 1.087, 1.058, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
7	[[6], [1.0875, 1.0125, 1.1], [1.1, 1.086, 1.057, 1.1, 1.1]]	[[18], [0.9875, 1.0875, 1.0375], [1.1, 1.086, 1.057, 1.056, 1.096]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.0375, 0.9, 0.9875], [1.1, 1.086, 1.057, 1.093, 1.1]]

8	[[12], [0.9625, 1.1, 1.025], [1.1, 1.086, 1.097, 1.1, 1.1]]	[[18], [1.05, 0.9, 1.0], [1.1, 1.086, 1.057, 1.085, 1.094]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
9	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.0375, 0.9375, 1.0], [1.1, 1.086, 1.057, 1.097, 1.095]]	[[18], [1.1, 0.9, 1.0], [1.1, 1.087, 1.058, 1.1, 1.099]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
10	[[0], [0.9625, 1.1, 1.0375], [1.1, 1.085, 1.1, 1.089, 1.1]]	[[18], [1.05, 0.9, 1.0], [1.1, 1.086, 1.058, 1.083, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
11	[[18], [0.9375, 1.0875, 1.0], [1.1, 1.1, 1.056, 1.1, 1.1]]	[[18], [1.05, 0.9, 1.0], [1.1, 1.086, 1.057, 1.09, 1.099]]	[[18], [1.1, 0.9, 1.0], [1.1, 1.087, 1.058, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
12	[[18], [0.9875, 0.9375, 0.975], [1.1, 1.1, 1.055, 1.1, 1.1]]	[[18], [1.0375, 0.9, 0.9875], [1.1, 1.086, 1.057, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
13	[[18], [0.9625, 1.1, 1.0125], [1.1, 1.086, 1.056, 1.1, 1.1]]	[[18], [1.0375, 0.9, 0.9875], [1.1, 1.086, 1.057, 1.093, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
14	[[12], [1.0625, 1.1, 1.1], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[18], [1.0, 1.0125, 1.0125], [1.1, 1.087, 1.058, 1.067, 1.098]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
15	[[12], [1.05, 1.1, 1.1], [1.1, 1.086, 1.1, 1.003, 1.092]]	[[18], [1.0625, 0.9, 0.9875], [1.1, 1.086, 1.057, 1.088, 1.098]]	[[18], [1.0375, 0.9, 0.9875], [1.1, 1.086, 1.057, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
16	[[18], [1.0125, 0.9, 0.9875], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[18], [1.0875, 0.9, 1.0], [1.1, 1.087, 1.058, 1.092, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
17	[[18], [0.9625, 1.1, 1.0125], [1.1, 1.086, 1.056, 1.076, 1.1]]	[[18], [1.025, 0.9, 0.9875], [1.1, 1.086, 1.1, 1.1, 1.098]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
18	[[0], [1.0375, 0.9, 1.0], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[18], [0.9875, 0.9625, 0.9875], [1.1, 1.086, 1.056, 1.096, 1.1]]	[[18], [1.1, 0.9, 1.0], [1.1, 1.087, 1.058, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
19	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.925, 0.9875], [1.1, 1.086, 1.057, 1.091, 1.099]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]
20	[[12], [1.0375, 0.9, 0.9875], [1.1, 1.086, 1.056, 1.1, 1.1]]	[[18], [1.0, 0.9375, 0.9875], [1.1, 1.086, 1.057, 1.097, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]

In Table 14 are displayed the optimal solutions (control variables) in each run, as [[Q_{c1}, Q_{c2}], [T_{k1}], [V_{g1}, V_{g2}, V_{g3}, $V_{g4,}$ $V_{g5,}$ V_{g6}, V_{g7}, V_{g8}, $V_{g9,}$ V_{g10}]] in [[Mvar], [p.u], [p.u]] for the 39 bus system:

Table 14 - Optimal solutions for IEEE 39 bus, in [[Mvar], [p.u], [p.u]]

Run no.	PSO	FDR PSO	SOPSO	FDR SOPSO
1	[[25, 50], [0.95], [1.041, 1.1, 1.037, 1.042, 1.051, 1.094, 1.1, 1.072, 1.07, 1.078]]	[[100, 125], [0.975], [1.04, 1.093, 1.03, 1.042, 1.043, 1.094, 1.097, 1.061, 1.074, 1.066]]	[[50, 25], [0.9], [1.037, 1.041, 1.053, 1.043, 1.048, 1.097, 1.097, 1.07, 1.07, 1.076]]	[[100, 100], [0.9], [1.032, 1.02, 1.051, 1.043, 1.047, 1.095, 1.1, 1.069, 1.07, 1.078]]
2	[[100, 100], [0.9375], [1.041, 1.055, 1.041, 1.04, 1.048, 1.1, 1.09, 1.073, 1.069, 1.068]]	[[100, 100], [0.925], [1.045, 1.036, 1.042, 1.029, 1.025, 1.097, 1.1, 1.074, 1.072, 1.069]]	[[75, 150], [0.9], [1.037, 1.01, 1.053, 1.042, 1.048, 1.095, 1.1, 1.073, 1.067, 1.079]]	[[100, 100], [0.9125], [1.035, 1.034, 1.052, 1.042, 1.049, 1.095, 1.1, 1.071, 1.068, 1.077]]
3	[[50, 75], [0.9625], [1.04, 1.1, 1.044, 1.02, 1.031, 1.1, 1.1, 1.075, 1.073, 1.074]]	[[75, 75], [0.9125], [1.045, 1.053, 1.038, 1.046, 1.04, 1.097, 1.091, 1.08, 1.062, 1.072]]	[[25, 150], [0.9], [1.038, 1.021, 1.049, 1.042, 1.05, 1.096, 1.1, 1.071, 1.069, 1.079]]	[[100, 125], [0.9125], [1.035, 1.029, 1.046, 1.043, 1.046, 1.095, 1.1, 1.071, 1.069, 1.077]]
4	[[25, 100], [0.925], [1.035, 1.068, 1.049, 1.043, 1.046, 1.095, 1.1, 1.07, 1.07, 1.08]]	[[25, 75], [0.9625], [1.047, 1.096, 1.051, 1.031, 1.04, 1.094, 1.1, 1.074, 1.069, 1.079]]	[[100, 100], [0.9], [1.035, 1.024, 1.045, 1.041, 1.05, 1.095, 1.1, 1.07, 1.069, 1.078]]	[[100, 125], [0.925], [1.033, 1.045, 1.044, 1.042, 1.048, 1.095, 1.1, 1.068, 1.071, 1.075]]
5	[[25, 75], [0.9875], [1.044, 1.1, 1.055, 1.03, 1.03, 1.1, 1.1, 1.077, 1.073, 1.069]]	[[75, 150], [0.9625], [1.039, 1.076, 1.04, 1.036, 1.036, 1.097, 1.099, 1.073, 1.07, 1.078]]	[[100, 0], [0.95], [1.036, 1.1, 1.048, 1.043, 1.047, 1.095, 1.1, 1.071, 1.068, 1.078]]	[[100, 150], [0.9125], [1.036, 1.02, 1.05, 1.042, 1.049, 1.095, 1.1, 1.071, 1.068, 1.077]]
6	[[50, 50], [0.9375], [1.041, 1.084, 1.05, 1.044, 1.045, 1.096, 1.097, 1.079, 1.062, 1.085]]	[[100, 150], [0.9875], [1.03, 1.095, 1.057, 1.048, 1.035, 1.089, 1.099, 1.075, 1.072, 1.067]]	[[75, 125], [0.975], [1.037, 1.1, 1.052, 1.042, 1.05, 1.097, 1.096, 1.072, 1.068, 1.077]]	[[100, 100], [0.95], [1.037, 1.075, 1.053, 1.043, 1.047, 1.095, 1.1, 1.072, 1.066, 1.079]]
7	[[75, 175], [0.9], [1.043,	[[100, 150], [0.9625], [1.046,	[[100, 50], [0.9], [1.038, 1.023, 1.054, 1.042, 1.049,	[[100, 100], [0.9125], [1.035,

	1.008, 1.043, 1.041, 1.05, 1.095, 1.1, 1.075, 1.064, 1.077]]	1.055, 1.058, 1.047, 1.034, 1.091, 1.089, 1.077, 1.067, 1.065]]	1.097, 1.095, 1.074, 1.066, 1.079]]	1.038, 1.046, 1.042, 1.049, 1.095, 1.1, 1.071, 1.068, 1.078]]
8	[[25, 50], [0.9], [1.042, 1.047, 1.047, 1.041, 1.053, 1.093, 1.1, 1.078, 1.066, 1.07]]	[[75, 125], [0.9125], [1.04, 1.029, 1.051, 1.041, 1.045, 1.091, 1.096, 1.066, 1.073, 1.073]]	[[50, 75], [0.925], [1.034, 1.053, 1.055, 1.043, 1.048, 1.097, 1.095, 1.069, 1.072, 1.078]]	[[100, 100], [0.9], [1.035, 1.022, 1.047, 1.042, 1.05, 1.095, 1.1, 1.071, 1.068, 1.077]]
9	[[25, 175], [0.9], [1.037, 1.005, 1.055, 1.043, 1.047, 1.098, 1.094, 1.071, 1.07, 1.079]]	[[75, 50], [0.9], [1.041, 1.04, 1.053, 1.037, 1.039, 1.092, 1.091, 1.068, 1.072, 1.069]]	[[50, 100], [0.9], [1.035, 1.026, 1.053, 1.042, 1.049, 1.096, 1.1, 1.072, 1.07, 1.076]]	[[100, 100], [0.9], [1.034, 1.015, 1.05, 1.042, 1.048, 1.095, 1.1, 1.07, 1.07, 1.077]]
10	[[0, 100], [0.9], [1.046, 0.956, 1.1, 1.036, 1.05, 1.097, 1.1, 1.072, 1.071, 1.078]]	[[50, 100], [0.9125], [1.031, 1.028, 1.069, 1.04, 1.043, 1.091, 1.063, 1.066, 1.075, 1.066]]	[[25, 175], [0.925], [1.038, 1.04, 1.048, 1.042, 1.05, 1.096, 1.1, 1.073, 1.069, 1.076]]	[[75, 100], [0.9125], [1.038, 1.038, 1.051, 1.043, 1.047, 1.095, 1.1, 1.074, 1.066, 1.08]]
11	[[50, 25], [0.9375], [1.037, 1.089, 1.051, 1.045, 1.042, 1.095, 1.1, 1.074, 1.067, 1.08]]	[[75, 75], [0.9125], [1.051, 1.049, 1.026, 1.036, 1.038, 1.091, 1.1, 1.071, 1.072, 1.059]]	[[75, 50], [0.9], [1.036, 1.031, 1.053, 1.041, 1.051, 1.095, 1.1, 1.069, 1.07, 1.076]]	[[100, 150], [0.9375], [1.035, 1.052, 1.048, 1.042, 1.047, 1.095, 1.1, 1.072, 1.068, 1.077]]
12	[[50, 125], [0.925], [1.036, 1.059, 1.044, 1.042, 1.049, 1.095, 1.1, 1.075, 1.067, 1.078]]	[[50, 175], [0.9], [1.045, 1.007, 1.048, 1.045, 1.037, 1.096, 1.099, 1.072, 1.069, 1.062]]	[[100, 150], [0.95], [1.032, 1.062, 1.051, 1.041, 1.05, 1.095, 1.1, 1.068, 1.071, 1.08]]	[[100, 125], [0.925], [1.039, 1.043, 1.045, 1.041, 1.05, 1.095, 1.1, 1.076, 1.064, 1.079]]
13	[[100, 50], [0.9], [1.036, 1.031, 1.047, 1.042, 1.05, 1.094, 1.1, 1.074, 1.067, 1.076]]	[[75, 125], [0.9], [1.043, 1.009, 1.051, 1.042, 1.033, 1.096, 1.095, 1.074, 1.069, 1.065]]	[[100, 100], [0.9], [1.033, 1.02, 1.049, 1.042, 1.049, 1.095, 1.1, 1.07, 1.069, 1.078]]	[[100, 125], [0.9375], [1.033, 1.054, 1.047, 1.042, 1.048, 1.096, 1.1, 1.068, 1.071, 1.075]]
14	[[50, 150], [0.9625], [1.04, 1.027, 1.086, 1.04, 1.053, 1.096, 1.1,	[[75, 150], [0.9], [1.049, 0.998, 1.04, 1.039, 1.051, 1.093, 1.099, 1.077, 1.065, 1.072]]	[[25, 25], [0.9375], [1.038, 1.099, 1.049, 1.043, 1.048, 1.096, 1.1, 1.071, 1.069, 1.077]]	[[100, 100], [0.9125], [1.037, 1.037, 1.046, 1.042, 1.049, 1.095, 1.1, 1.072, 1.067, 1.079]]

	1.069, 1.072, 1.073]]			
15	[[75, 0], [0.9375], [1.042, 1.096, 1.05, 1.044, 1.043, 1.094, 1.1, 1.075, 1.064, 1.074]]	[[50, 125], [0.9], [1.04, 1.025, 1.049, 1.042, 1.051, 1.094, 1.093, 1.07, 1.07, 1.074]]	[[75, 200], [0.9875], [1.038, 1.091, 1.055, 1.043, 1.047, 1.095, 1.1, 1.073, 1.067, 1.077]]	[[100, 125], [0.95], [1.035, 1.068, 1.049, 1.041, 1.05, 1.095, 1.1, 1.071, 1.068, 1.077]]
16	[[100, 125], [0.9], [1.04, 1.005, 1.035, 1.031, 1.032, 1.1, 1.097, 1.07, 1.073, 1.074]]	[[75, 125], [0.925], [1.04, 1.054, 1.036, 1.044, 1.042, 1.092, 1.098, 1.072, 1.071, 1.068]]	[[0, 100], [0.9625], [1.04, 1.099, 1.052, 1.043, 1.048, 1.096, 1.1, 1.072, 1.07, 1.076]]	[[100, 200], [0.9], [1.037, 0.991, 1.052, 1.043, 1.046, 1.095, 1.1, 1.074, 1.066, 1.079]]
17	[[25, 100], [0.9375], [1.038, 1.07, 1.045, 1.043, 1.049, 1.1, 1.09, 1.07, 1.071, 1.08]]	[[100, 100], [0.925], [1.049, 1.057, 1.023, 1.038, 1.056, 1.083, 1.099, 1.078, 1.066, 1.064]]	[[50, 25], [0.9], [1.035, 1.047, 1.051, 1.042, 1.05, 1.096, 1.1, 1.069, 1.071, 1.077]]	[[100, 100], [0.9], [1.037, 1.02, 1.048, 1.043, 1.047, 1.095, 1.1, 1.073, 1.066, 1.077]]
18	[[100, 125], [0.975], [1.041, 1.1, 1.042, 1.042, 1.044, 1.1, 1.088, 1.071, 1.069, 1.07]]	[[100, 75], [0.95], [1.047, 1.055, 1.069, 1.042, 1.053, 1.076, 1.1, 1.073, 1.069, 1.064]]	[[25, 125], [0.9125], [1.036, 1.041, 1.052, 1.044, 1.046, 1.096, 1.1, 1.069, 1.071, 1.075]]	[[100, 100], [0.9625], [1.035, 1.088, 1.048, 1.043, 1.047, 1.095, 1.1, 1.071, 1.069, 1.078]]
19	[[50, 100], [0.9625], [1.034, 1.1, 1.054, 1.042, 1.05, 1.095, 1.1, 1.072, 1.07, 1.075]]	[[50, 100], [0.9125], [1.04, 1.036, 1.057, 1.038, 1.044, 1.091, 1.092, 1.075, 1.071, 1.061]]	[[50, 125], [0.9], [1.035, 1.019, 1.055, 1.042, 1.048, 1.095, 1.1, 1.071, 1.069, 1.08]]	[[100, 100], [0.9125], [1.035, 1.037, 1.046, 1.043, 1.047, 1.095, 1.1, 1.072, 1.068, 1.078]]
20	[[0, 0], [0.9], [1.034, 1.074, 1.04, 1.042, 1.051, 1.1, 1.089, 1.07, 1.072, 1.078]]	[[100, 125], [0.95], [1.025, 1.098, 1.025, 1.044, 1.046, 1.093, 1.098, 1.067, 1.073, 1.081]]	[[0, 50], [0.9], [1.04, 1.046, 1.052, 1.043, 1.048, 1.098, 1.095, 1.073, 1.068, 1.08]]	[[100, 50], [0.9375], [1.036, 1.073, 1.05, 1.043, 1.047, 1.096, 1.097, 1.072, 1.068, 1.078]]

A.2 – State Variables

In Table 15 are displayed the state variables in each run, as

$[Q_{G_1}, Q_{G_2}, \dots, Q_{G_5}], [V_1, V_2, \dots, V_{14}], [S_{l_1}, S_{l_2}, \dots, S_{l_{20}}]$ in [[Mvar], [p.u], [MVA]] for the 14 bus system:

Table 15 - State variables for IEEE 14 bus, in [[Mvar], [p.u], [MVA]]

Run no.	PSO	FDR PSO
1	[-22.896, 22.928, 40.0, 24.0, 8.79], [1.1, 1.086, 1.07, 1.066, 1.071, 1.097, 1.086, 1.1, 1.096, 1.089, 1.089, 1.083, 1.08, 1.071], [157.37, 75.485, 154.38, 73.994, 55.979, 41.134, 72.403, 24.748, 54.345, 26.422, 63.237, 32.233, 26.612, 73.221, 40.187, 63.701, 42.313, 42.797, 6.367, 7.882]	[-21.852, 32.966, 26.001, 24.0, 10.683], [1.1, 1.086, 1.057, 1.064, 1.069, 1.09, 1.083, 1.1, 1.083, 1.077, 1.08, 1.076, 1.072, 1.061], [157.285, 75.515, 154.3, 73.089, 56.107, 41.299, 71.04, 23.498, 54.488, 24.448, 62.996, 29.852, 18.82, 73.169, 40.392, 63.459, 42.715, 43.031, 6.921, 7.99]
2	[-23.083, 25.561, 40.0, 24.0, 24.0], [1.1, 1.086, 1.07, 1.065, 1.07, 0.985, 0.999, 1.039, 0.985, 0.977, 0.978, 0.97, 0.966, 0.957], [157.667, 75.396, 154.694, 74.063, 56.048, 40.997, 72.442, 24.815, 54.424, 26.522, 64.031, 30.46, 22.273, 73.105, 40.079, 64.503, 41.26, 41.794, 5.744, 7.831]	[-26.086, 28.978, 24.989, 24.0, 24.0], [1.1, 1.087, 1.06, 1.07, 1.073, 1.057, 1.049, 1.088, 1.054, 1.047, 1.049, 1.043, 1.039, 1.029], [157.419, 75.734, 154.511, 72.951, 56.483, 41.455, 70.974, 23.413, 54.811, 24.135, 63.969, 37.371, 23.57, 73.545, 40.474, 64.434, 42.377, 42.802, 6.538, 7.93]
3	[-22.876, 22.906, 40.0, 24.0, 8.793], [1.1, 1.086, 1.07, 1.066, 1.071, 1.097, 1.086, 1.1, 1.096, 1.089, 1.089, 1.083, 1.08, 1.071], [157.367, 75.485, 154.376, 73.994, 55.979, 41.134, 72.403, 24.748, 54.345, 26.421, 63.237, 32.234, 26.611, 73.221, 40.187, 63.701, 42.313, 42.797, 6.367, 7.882]	[-23.387, 31.725, 25.895, 24.0, 15.028], [1.1, 1.086, 1.058, 1.065, 1.07, 1.083, 1.076, 1.1, 1.081, 1.074, 1.075, 1.069, 1.065, 1.057], [157.342, 75.565, 154.383, 73.072, 56.218, 41.307, 71.048, 23.45, 54.585, 24.352, 63.493, 33.211, 23.449, 73.281, 40.372, 63.961, 42.274, 42.759, 6.354, 7.885]
4	[-37.156, 64.0, 40.0, 23.669, -18.0], [1.1, 1.095, 1.07, 1.058, 1.066, 1.1, 1.098, 1.068, 1.093, 1.087, 1.09, 1.086, 1.082, 1.071], [161.338, 74.825, 158.97, 74.095, 55.871, 41.252, 72.182, 26.845, 54.436, 28.831, 62.07, 39.105, 16.466, 72.418, 40.61, 62.529, 42.431, 42.691, 6.829, 7.978]	[-21.682, 32.999, 26.126, 24.0, 11.42], [1.1, 1.086, 1.057, 1.064, 1.068, 1.075, 1.082, 1.1, 1.067, 1.061, 1.064, 1.061, 1.056, 1.044], [157.299, 75.512, 154.31, 73.097, 56.108, 41.298, 71.051, 23.499, 54.49, 24.458, 62.991, 30.114, 17.341, 73.157, 40.395, 63.453, 42.761, 43.027, 7.062, 8.019]
5	[-22.642, 34.077, 26.649, 24.0, 24.0], [1.1, 1.087, 1.058, 1.064, 1.068, 1.06, 1.036, 1.075, 1.049, 1.043, 1.048, 1.045, 1.04, 1.027], [157.464, 75.497, 154.509, 73.121, 56.093, 41.288, 71.091, 23.518, 54.477, 24.516, 62.95, 41.016, 34.527, 73.146, 40.395, 63.411, 42.896, 43.074, 7.339, 8.067]	[-23.999, 30.99, 25.521, 24.0, 20.389], [1.1, 1.087, 1.058, 1.067, 1.071, 1.079, 1.058, 1.091, 1.073, 1.067, 1.069, 1.065, 1.061, 1.05], [157.348, 75.621, 154.403, 73.01, 56.263, 41.375, 71.0, 23.445, 54.618, 24.283, 63.348, 40.708, 30.783, 73.352, 40.433, 63.811, 42.714, 43.044, 6.895, 7.987]
6	[-37.736, 64.0, 21.525, 24.0, 24.0], [1.1, 1.094, 1.056, 1.062, 1.07, 0.985, 1.009, 1.05, 0.984, 0.976, 0.977, 0.969, 0.965, 0.956],	[-23.511, 31.873, 25.271, 24.0, 14.72], [1.1, 1.087, 1.057, 1.066, 1.07, 1.08, 1.076, 1.1, 1.075, 1.069, 1.071, 1.066,

	[160.712, 75.135, 158.302, 73.389, 56.238, 41.182, 71.037, 23.469, 54.722, 24.469, 63.883, 30.14, 19.01, 72.842, 40.45, 64.363, 41.082, 41.596, 5.67, 7.823]	1.062, 1.051], [157.343, 75.587, 154.389, 73.026, 56.231, 41.343, 70.991, 23.445, 54.594, 24.295, 63.356, 30.678, 18.53, 73.3, 40.409, 63.821, 42.54, 42.92, 6.707, 7.956]
7	[-22.093, 36.76, 27.895, 24.0, 24.0], [1.1, 1.086, 1.057, 1.061, 1.067, 0.982, 0.995, 1.036, 0.982, 0.974, 0.974, 0.967, 0.963, 0.954], [157.703, 75.357, 154.742, 73.329, 56.136, 41.1, 71.274, 23.6, 54.543, 24.762, 63.929, 30.452, 22.204, 72.971, 40.237, 64.407, 41.269, 41.785, 5.774, 7.839]	[-23.23, 30.841, 25.487, 24.0, 18.066], [1.1, 1.086, 1.057, 1.066, 1.071, 1.045, 1.067, 1.096, 1.044, 1.037, 1.037, 1.03, 1.027, 1.018], [157.317, 75.586, 154.346, 73.062, 56.306, 41.312, 71.036, 23.414, 54.664, 24.269, 63.837, 31.415, 18.992, 73.309, 40.369, 64.307, 41.973, 42.49, 6.16, 7.866]
8	[-21.697, 24.789, 40.0, 24.0, 15.604], [1.1, 1.086, 1.069, 1.064, 1.069, 1.051, 1.075, 1.1, 1.044, 1.038, 1.041, 1.036, 1.032, 1.021], [157.438, 75.415, 154.432, 74.002, 55.922, 41.076, 72.383, 24.872, 54.302, 26.574, 63.15, 35.146, 19.867, 73.091, 40.163, 63.613, 42.312, 42.632, 6.686, 7.97]	[-22.762, 31.45, 25.999, 24.0, 16.625], [1.1, 1.086, 1.057, 1.065, 1.07, 1.081, 1.067, 1.094, 1.079, 1.072, 1.074, 1.068, 1.064, 1.055], [157.285, 75.559, 154.31, 73.077, 56.202, 41.308, 71.052, 23.458, 54.57, 24.371, 63.409, 37.207, 29.272, 73.261, 40.377, 63.876, 42.347, 42.805, 6.441, 7.903]
9	[-19.663, 31.947, 25.623, 23.253, 12.291], [1.1, 1.085, 1.055, 1.063, 1.067, 1.1, 1.08, 1.1, 1.092, 1.086, 1.09, 1.086, 1.082, 1.07], [157.071, 75.512, 154.027, 73.072, 56.088, 41.319, 71.005, 23.501, 54.471, 24.432, 62.821, 34.542, 26.379, 73.122, 40.417, 63.283, 42.934, 43.143, 7.142, 8.022]	[-22.746, 30.577, 24.943, 24.0, 16.963], [1.1, 1.086, 1.057, 1.066, 1.07, 1.079, 1.068, 1.095, 1.073, 1.067, 1.069, 1.065, 1.061, 1.05], [157.215, 75.614, 154.234, 72.995, 56.265, 41.377, 70.961, 23.442, 54.622, 24.249, 63.359, 33.863, 22.353, 73.323, 40.435, 63.823, 42.64, 42.983, 6.825, 7.976]
10	[-18.934, 29.463, 40.0, 24.0, 22.515], [1.1, 1.085, 1.066, 1.06, 1.065, 1.032, 1.064, 1.1, 1.023, 1.017, 1.021, 1.017, 1.013, 1.0], [157.511, 75.27, 154.464, 74.019, 55.78, 40.994, 72.316, 25.299, 54.203, 27.077, 62.846, 36.903, 18.091, 72.82, 40.147, 63.312, 42.2, 42.427, 6.83, 8.004]	[-23.547, 30.938, 25.361, 24.0, 18.458], [1.1, 1.086, 1.058, 1.066, 1.071, 1.083, 1.07, 1.1, 1.082, 1.075, 1.076, 1.07, 1.066, 1.057], [157.296, 75.604, 154.334, 73.031, 56.267, 41.345, 71.005, 23.434, 54.626, 24.278, 63.551, 38.024, 29.025, 73.335, 40.399, 64.019, 42.316, 42.812, 6.372, 7.886]
11	[-37.973, 64.0, 20.555, 24.0, 4.303], [1.1, 1.095, 1.056, 1.063, 1.07, 1.076, 1.093, 1.1, 1.068, 1.063, 1.066, 1.062, 1.058, 1.046], [160.595, 75.208, 158.196, 73.268, 56.069, 41.33, 70.907, 23.513, 54.556, 24.43, 62.616, 38.254, 21.505, 72.913, 40.591, 63.078, 42.614, 42.884, 6.97, 8.007]	[-23.148, 30.655, 25.429, 24.0, 18.351], [1.1, 1.086, 1.057, 1.066, 1.071, 1.083, 1.07, 1.099, 1.082, 1.075, 1.075, 1.069, 1.066, 1.057], [157.253, 75.599, 154.28, 73.035, 56.263, 41.343, 71.009, 23.435, 54.622, 24.285, 63.543, 37.963, 29.042, 73.323, 40.397, 64.01, 42.31, 42.799, 6.371, 7.886]

12	[-38.799, 64.0, 18.931, 21.465, 5.537], [1.1, 1.095, 1.055, 1.065, 1.07, 1.1, 1.091, 1.1, 1.093, 1.087, 1.09, 1.086, 1.082, 1.071], [160.668, 75.247, 158.29, 73.203, 56.095, 41.348, 70.821, 23.486, 54.569, 24.255, 62.713, 28.963, 19.454, 72.974, 40.601, 63.174, 42.756, 42.845, 6.954, 7.992]	[-22.622, 31.971, 25.708, 24.0, 15.179], [1.1, 1.086, 1.057, 1.065, 1.07, 1.093, 1.076, 1.1, 1.088, 1.081, 1.084, 1.079, 1.075, 1.064], [157.275, 75.557, 154.301, 73.053, 56.167, 41.325, 71.017, 23.469, 54.538, 24.369, 63.165, 36.234, 27.7, 73.246, 40.4, 63.629, 42.634, 43.006, 6.762, 7.959]
13	[-21.361, 33.461, 26.244, 24.0, 13.004], [1.1, 1.086, 1.056, 1.063, 1.068, 1.062, 1.079, 1.1, 1.054, 1.048, 1.051, 1.047, 1.043, 1.031], [157.318, 75.491, 154.323, 73.124, 56.1, 41.276, 71.071, 23.507, 54.487, 24.49, 63.048, 33.56, 21.08, 73.123, 40.38, 63.512, 42.617, 42.902, 6.934, 8.003]	[-22.841, 32.116, 25.881, 24.0, 15.002], [1.1, 1.086, 1.057, 1.065, 1.07, 1.093, 1.076, 1.1, 1.088, 1.081, 1.084, 1.079, 1.075, 1.065], [157.307, 75.556, 154.339, 73.061, 56.162, 41.323, 71.03, 23.474, 54.533, 24.385, 63.152, 36.171, 27.72, 73.247, 40.398, 63.616, 42.647, 43.02, 6.777, 7.962]
14	[-24.435, 20.373, 40.0, 24.0, 24.0], [1.1, 1.086, 1.072, 1.069, 1.073, 0.987, 1.008, 1.048, 0.986, 0.978, 0.979, 0.972, 0.968, 0.958], [157.514, 75.578, 154.546, 74.068, 56.293, 41.172, 72.527, 24.446, 54.625, 26.065, 64.35, 30.292, 16.591, 73.376, 40.195, 64.818, 41.528, 42.027, 5.95, 7.864]	[-24.049, 31.341, 26.049, 24.0, 15.371], [1.1, 1.087, 1.058, 1.066, 1.071, 1.067, 1.074, 1.098, 1.063, 1.057, 1.059, 1.053, 1.049, 1.039], [157.401, 75.592, 154.459, 73.059, 56.254, 41.33, 71.055, 23.443, 54.613, 24.331, 63.507, 29.517, 16.535, 73.321, 40.392, 63.972, 42.405, 42.805, 6.593, 7.941]
15	[-24.339, 20.164, 40.0, 24.0, 24.0], [1.1, 1.086, 1.072, 1.069, 1.073, 0.99, 1.016, 1.056, 0.991, 0.983, 0.983, 0.975, 0.971, 0.963], [157.49, 75.575, 154.517, 74.077, 56.303, 41.163, 72.533, 24.447, 54.636, 26.069, 64.47, 30.664, 16.703, 73.378, 40.181, 64.941, 41.309, 41.913, 5.695, 7.807]	[-22.775, 31.031, 25.13, 22.623, 19.94], [1.1, 1.086, 1.057, 1.066, 1.07, 1.088, 1.066, 1.098, 1.08, 1.074, 1.077, 1.074, 1.069, 1.058], [157.257, 75.596, 154.284, 72.996, 56.247, 41.372, 70.968, 23.449, 54.604, 24.269, 63.274, 39.992, 29.247, 73.291, 40.441, 63.735, 42.905, 43.074, 7.085, 8.015]
16	[-22.978, 23.016, 40.0, 24.0, 8.78], [1.1, 1.086, 1.07, 1.066, 1.071, 1.097, 1.086, 1.1, 1.096, 1.089, 1.089, 1.083, 1.08, 1.071], [157.38, 75.484, 154.393, 73.994, 55.978, 41.133, 72.403, 24.749, 54.344, 26.423, 63.237, 32.231, 26.614, 73.222, 40.187, 63.701, 42.313, 42.798, 6.368, 7.882]	[-24.167, 29.482, 24.541, 24.0, 24.0], [1.1, 1.087, 1.058, 1.068, 1.072, 1.079, 1.056, 1.094, 1.072, 1.066, 1.069, 1.065, 1.061, 1.049], [157.257, 75.693, 154.307, 72.93, 56.369, 41.452, 70.92, 23.444, 54.708, 24.17, 63.48, 43.322, 31.272, 73.448, 40.49, 63.942, 42.853, 43.15, 7.042, 8.011]
17	[-21.01, 33.251, 25.974, 24.0, 13.041], [1.1, 1.086, 1.056, 1.063, 1.068, 1.062, 1.079, 1.1, 1.054, 1.048, 1.051, 1.047, 1.043, 1.031], [157.262, 75.492, 154.257, 73.11, 56.102, 41.281, 71.049, 23.499, 54.488, 24.464, 63.038, 33.53, 21.085, 73.12, 40.384, 63.502, 42.62, 42.912, 6.94, 8.003]	[-23.39, 22.013, 40.0, 24.0, 10.806], [1.1, 1.086, 1.071, 1.067, 1.071, 1.096, 1.081, 1.098, 1.092, 1.086, 1.087, 1.082, 1.078, 1.069], [157.364, 75.526, 154.383, 73.99, 56.027, 41.179, 72.418, 24.663, 54.382, 26.316, 63.223, 33.703, 27.392, 73.281,

		40.222, 63.686, 42.498, 42.914, 6.592, 7.927]
18	[-21.324, 30.199, 40.0, 24.0, 24.0], [1.1, 1.086, 1.068, 1.062, 1.067, 1.07, 1.061, 1.1, 1.059, 1.053, 1.058, 1.056, 1.051, 1.037], [157.668, 75.298, 154.688, 73.985, 55.744, 41.037, 72.303, 25.229, 54.162, 26.994, 62.518, 32.864, 31.905, 72.893, 40.183, 62.978, 42.757, 42.901, 7.362, 8.07]	[-20.929, 33.328, 26.216, 24.0, 8.871], [1.1, 1.086, 1.056, 1.063, 1.068, 1.09, 1.086, 1.1, 1.083, 1.077, 1.08, 1.076, 1.071, 1.061], [157.241, 75.484, 154.235, 73.12, 56.069, 41.277, 71.06, 23.522, 54.459, 24.511, 62.933, 29.056, 17.484, 73.112, 40.38, 63.397, 42.663, 42.99, 6.873, 7.982]
19	[-19.708, 32.014, 25.609, 23.247, 12.289], [1.1, 1.085, 1.055, 1.063, 1.067, 1.1, 1.08, 1.1, 1.092, 1.086, 1.09, 1.086, 1.082, 1.07], [157.077, 75.511, 154.035, 73.072, 56.087, 41.318, 71.004, 23.501, 54.471, 24.431, 62.821, 34.541, 26.38, 73.121, 40.417, 63.283, 42.934, 43.142, 7.141, 8.022]	[-21.934, 31.468, 25.259, 24.0, 14.161], [1.1, 1.086, 1.057, 1.065, 1.069, 1.091, 1.077, 1.099, 1.084, 1.078, 1.081, 1.077, 1.072, 1.061], [157.193, 75.572, 154.199, 73.025, 56.178, 41.352, 70.979, 23.465, 54.546, 24.327, 63.085, 33.242, 22.86, 73.249, 40.426, 63.548, 42.799, 43.11, 6.996, 8.001]
20	[-20.814, 33.891, 26.477, 24.0, 18.418], [1.1, 1.086, 1.056, 1.062, 1.067, 1.087, 1.071, 1.1, 1.078, 1.072, 1.076, 1.073, 1.068, 1.056], [157.282, 75.471, 154.278, 73.134, 56.042, 41.278, 71.076, 23.545, 54.436, 24.56, 62.792, 35.233, 28.75, 73.082, 40.39, 63.254, 42.842, 43.078, 7.177, 8.034]	[-21.614, 33.277, 26.282, 24.0, 9.895], [1.1, 1.086, 1.057, 1.063, 1.068, 1.091, 1.084, 1.1, 1.087, 1.08, 1.082, 1.077, 1.073, 1.063], [157.288, 75.492, 154.297, 73.119, 56.088, 41.273, 71.068, 23.514, 54.475, 24.499, 63.043, 30.117, 20.133, 73.139, 40.37, 63.508, 42.558, 42.931, 6.714, 7.953]

In Table 16 are displayed the state variables in each run, as $[Q_{G_1}, Q_{G_2}, \dots, Q_{G_{10}}], [V_1, V_2, \dots, V_{39}], [S_{l_1}, S_{l_2}, \dots, S_{l_{34}}]$ in [[Mvar], [p.u], [MVA]] for the 39 bus system:

Table 16 - State variables for IEEE 39 bus, in [[Mvar], [p.u], [MVA]]

Run no.	PSO	FDR PSO
1	[-80.079, 340.693, 128.866, 99.818, 142.259, 195.537, 55.488, 29.737, -0.215, 32.459], [1.09, 1.082, 1.081, 1.077, 1.089, 1.092, 1.08, 1.078, 1.091, 1.091, 1.09, 1.078, 1.088, 1.083, 1.076, 1.087, 1.087, 1.084, 1.1, 1.035, 1.084, 1.099, 1.093, 1.091, 1.096, 1.1, 1.089, 1.098, 1.097, 1.041, 1.1, 1.037, 1.042, 1.051, 1.094, 1.1, 1.072, 1.07, 1.078], [118.663,	[-65.132, 196.497, 116.907, 116.214, 123.099, 204.894, 49.613, 1.344, 12.685, 14.296], [1.082, 1.078, 1.08, 1.079, 1.085, 1.085, 1.074, 1.072, 1.081, 1.086, 1.085, 1.073, 1.084, 1.081, 1.074, 1.085, 1.085, 1.082, 1.098, 1.031, 1.083, 1.097, 1.091, 1.089, 1.091, 1.099, 1.087, 1.1, 1.1, 1.04, 1.093, 1.03, 1.042, 1.043, 1.094, 1.097, 1.061, 1.074, 1.066], [119.632, 119.632, 131.495, 371.02, 238.117, 261.387, 368.094, 77.767, 42.151, 77.761, 168.286, 262.163,

	118.663, 138.162, 370.158, 237.868, 266.25, 367.723, 76.925, 41.876, 81.91, 186.589, 263.556, 181.096, 489.455, 326.55, 489.867, 440.39, 350.608, 614.625, 438.994, 192.879, 327.578, 193.947, 60.277, 20.222, 20.222, 350.0, 305.759, 651.712, 352.487, 349.481, 48.122, 47.274, 41.227, 306.495, 41.868, 294.349, 263.089, 294.931, 32.497, 60.63, 306.155, 302.939, 207.394, 453.788, 331.555]	165.276, 479.888, 331.996, 480.28, 438.547, 348.766, 571.736, 437.12, 191.845, 333.189, 192.637, 49.535, 14.348, 14.348, 349.601, 302.312, 650.924, 350.33, 349.22, 46.324, 45.531, 42.94, 302.637, 43.642, 293.876, 262.555, 293.318, 32.403, 60.926, 306.287, 303.13, 207.145, 453.476, 331.461]
2	[-77.662, 200.361, 146.784, 95.859, 135.141, 231.934, 21.782, 32.294, -2.007, 9.183], [1.084, 1.082, 1.083, 1.082, 1.088, 1.089, 1.077, 1.075, 1.084, 1.092, 1.089, 1.078, 1.089, 1.085, 1.077, 1.087, 1.087, 1.085, 1.098, 1.033, 1.085, 1.1, 1.091, 1.091, 1.096, 1.1, 1.089, 1.098, 1.097, 1.041, 1.055, 1.041, 1.04, 1.048, 1.1, 1.09, 1.073, 1.069, 1.068], [120.292, 120.292, 130.596, 371.723, 237.869, 265.404, 368.915, 78.068, 38.509, 77.967, 166.669, 262.203, 164.026, 479.369, 331.348, 479.757, 439.14, 347.268, 572.517, 437.736, 192.119, 332.569, 193.02, 48.599, 14.214, 14.214, 350.009, 303.918, 653.221, 348.401, 349.861, 45.933, 45.159, 43.304, 304.446, 44.012, 294.102, 262.427, 294.034, 35.209, 66.69, 303.955, 300.961, 207.836, 452.961, 331.631]	[-62.631, 186.396, 159.14, 94.636, 95.791, 217.845, 56.978, 31.789, 3.099, 13.687], [1.085, 1.083, 1.082, 1.08, 1.085, 1.086, 1.075, 1.073, 1.084, 1.09, 1.088, 1.076, 1.087, 1.083, 1.073, 1.083, 1.085, 1.083, 1.087, 1.017, 1.083, 1.099, 1.092, 1.088, 1.097, 1.1, 1.088, 1.099, 1.099, 1.045, 1.036, 1.042, 1.029, 1.025, 1.097, 1.1, 1.074, 1.072, 1.069], [120.388, 120.388, 130.699, 370.295, 237.978, 260.621, 367.914, 77.377, 34.546, 78.801, 166.117, 262.103, 163.673, 479.395, 330.722, 479.785, 438.491, 346.951, 569.776, 437.079, 191.793, 331.884, 192.591, 53.209, 15.915, 15.915, 350.38, 304.762, 654.499, 347.92, 350.311, 45.828, 45.055, 43.405, 305.361, 44.119, 294.35, 262.389, 294.475, 38.714, 72.11, 302.047, 299.308, 210.616, 452.265, 330.715]
3	[-82.653, 274.402, 159.769, 51.201, 126.602, 231.934, 54.81, 40.625, 6.244, 24.871], [1.088, 1.082, 1.081, 1.078, 1.088, 1.091, 1.079, 1.077, 1.088, 1.093, 1.091, 1.078, 1.089, 1.083, 1.073, 1.082, 1.084, 1.082, 1.084, 1.018, 1.083, 1.1, 1.093, 1.087, 1.096, 1.1, 1.087, 1.1, 1.1, 1.04, 1.1, 1.044, 1.02, 1.031, 1.1, 1.1, 1.075, 1.073, 1.074], [119.026, 119.026, 135.665, 370.625, 238.171, 267.178, 368.253, 77.072, 35.434, 79.384, 180.506, 262.693, 175.59, 483.547, 328.195, 483.935, 439.815, 348.071, 591.961, 438.419, 192.518, 329.304, 193.513, 56.201, 17.518, 17.518,	[-76.8, 303.0, 117.153, 128.866, 105.382, 212.82, 27.713, 47.182, -16.065, 2.597], [1.088, 1.085, 1.086, 1.085, 1.094, 1.096, 1.084, 1.082, 1.089, 1.095, 1.094, 1.082, 1.092, 1.088, 1.078, 1.088, 1.089, 1.086, 1.1, 1.03, 1.085, 1.099, 1.091, 1.092, 1.1, 1.1, 1.089, 1.094, 1.092, 1.045, 1.053, 1.038, 1.046, 1.04, 1.097, 1.091, 1.08, 1.062, 1.072], [120.204, 120.204, 130.918, 371.561, 238.137, 265.083, 368.914, 78.273, 34.778, 77.831, 176.835, 261.79, 172.265, 483.202, 330.204, 483.573, 441.249, 350.803, 601.326, 439.885, 193.391, 331.49, 194.605, 44.057, 14.245, 14.245, 350.037, 304.298, 651.01, 352.747, 349.489, 47.809, 46.978, 41.517, 304.918, 42.162, 293.894,

	349.78, 307.733, 654.612, 349.447, 349.527, 47.465, 46.64, 41.847, 308.603, 42.507, 295.311, 262.465, 296.229, 40.37, 74.384, 301.212, 298.563, 210.584, 452.894, 330.551]	261.987, 294.13, 38.542, 72.289, 302.072, 299.274, 208.222, 453.183, 332.24]
4	[-106.825, 293.442, 152.15, 108.095, 125.031, 195.801, 51.544, 26.229, 0.379, 27.863], [1.091, 1.08, 1.083, 1.083, 1.097, 1.099, 1.087, 1.085, 1.095, 1.099, 1.098, 1.085, 1.095, 1.089, 1.079, 1.088, 1.088, 1.085, 1.1, 1.033, 1.086, 1.1, 1.093, 1.093, 1.094, 1.1, 1.089, 1.099, 1.098, 1.035, 1.068, 1.049, 1.043, 1.046, 1.095, 1.1, 1.07, 1.07, 1.08], [118.355, 118.355, 140.882, 372.719, 237.747, 277.017, 369.59, 78.827, 41.556, 77.028, 196.907, 263.813, 190.602, 484.101, 328.653, 484.464, 440.331, 349.813, 598.208, 439.041, 192.886, 329.914, 193.994, 51.809, 15.008, 15.008, 349.998, 309.024, 653.881, 351.547, 349.583, 48.503, 47.653, 40.856, 310.044, 41.477, 295.838, 263.251, 297.17, 39.451, 73.795, 301.746, 299.011, 206.854, 453.148, 331.728]	[-59.778, 254.231, 186.261, 77.479, 131.222, 200.558, 60.013, 29.094, -2.087, 33.19], [1.092, 1.085, 1.082, 1.078, 1.088, 1.091, 1.079, 1.078, 1.091, 1.095, 1.092, 1.08, 1.09, 1.084, 1.075, 1.084, 1.086, 1.083, 1.091, 1.026, 1.083, 1.098, 1.091, 1.089, 1.098, 1.1, 1.088, 1.098, 1.097, 1.047, 1.096, 1.051, 1.031, 1.04, 1.094, 1.1, 1.074, 1.069, 1.079], [118.844, 118.844, 137.324, 369.033, 238.145, 259.802, 367.019, 76.99, 36.014, 82.584, 183.2, 264.378, 178.002, 483.708, 327.078, 484.096, 438.982, 347.3, 585.964, 437.617, 192.114, 328.126, 192.988, 63.483, 22.756, 22.756, 350.694, 309.876, 658.184, 348.279, 350.644, 47.096, 46.286, 42.196, 310.891, 42.864, 296.351, 263.764, 297.648, 38.047, 71.208, 302.376, 299.588, 209.622, 452.067, 331.14]
5	[-62.353, 186.28, 232.452, 91.449, 108.925, 231.934, 54.81, 46.504, 6.6, 26.92], [1.085, 1.082, 1.078, 1.071, 1.079, 1.08, 1.069, 1.067, 1.081, 1.089, 1.085, 1.073, 1.085, 1.078, 1.072, 1.082, 1.083, 1.08, 1.088, 1.019, 1.083, 1.1, 1.093, 1.087, 1.097, 1.1, 1.087, 1.1, 1.1, 1.044, 1.1, 1.055, 1.03, 1.03, 1.1, 1.1, 1.077, 1.073, 1.069], [120.243, 120.243, 131.943, 369.702, 238.423, 260.549, 367.901, 79.928, 40.495, 89.387, 172.225, 266.367, 168.497, 481.045, 327.909, 481.451, 438.521, 348.443, 569.711, 437.024, 191.767, 328.851, 192.539, 63.157, 23.181, 23.181, 355.923, 310.716, 666.359, 348.454, 356.299, 45.099, 44.347, 44.101, 311.713, 44.841, 296.853, 265.473, 298.062, 32.869, 60.315, 305.916, 302.736, 209.455, 452.04, 330.551]	[-91.113, 161.405, 141.356, 101.459, 111.136, 213.106, 50.257, 33.772, 0.49, 34.373], [1.091, 1.082, 1.082, 1.081, 1.09, 1.09, 1.079, 1.078, 1.091, 1.092, 1.09, 1.078, 1.089, 1.084, 1.076, 1.085, 1.086, 1.084, 1.093, 1.025, 1.084, 1.1, 1.093, 1.09, 1.096, 1.1, 1.088, 1.099, 1.098, 1.039, 1.076, 1.04, 1.036, 1.036, 1.097, 1.099, 1.073, 1.07, 1.078], [118.689, 118.689, 138.608, 371.087, 237.977, 270.382, 368.404, 77.598, 37.212, 77.973, 175.455, 262.105, 171.136, 480.836, 330.217, 481.23, 436.5, 347.946, 565.483, 435.205, 190.982, 331.433, 191.508, 61.549, 21.23, 21.23, 349.644, 304.596, 652.718, 349.326, 349.377, 46.667, 45.869, 42.606, 305.2, 43.29, 294.19, 262.254, 294.356, 36.887, 69.397, 302.936, 300.06, 208.746, 452.145, 331.076]
6	[-95.739, 304.844, 156.589, 108.546, 121.314, 201.191, 41.888, 42.649, -16.839, 38.122], [1.096, 1.085, 1.086,	[-121.273, 114.784, 205.177, 148.756, 90.734, 174.633, 58.777, 48.455, 4.227, 6.72], [1.082, 1.078, 1.082, 1.085, 1.091, 1.091, 1.079, 1.078,

	1.085, 1.095, 1.098, 1.087, 1.085, 1.098, 1.1, 1.098, 1.086, 1.096, 1.09, 1.08, 1.089, 1.089, 1.087, 1.1, 1.033, 1.086, 1.1, 1.093, 1.093, 1.099, 1.1, 1.09, 1.094, 1.092, 1.041, 1.084, 1.05, 1.044, 1.045, 1.096, 1.097, 1.079, 1.062, 1.085], [118.586, 118.586, 142.213, 371.062, 238.133, 272.211, 368.383, 77.722, 35.917, 77.652, 182.491, 263.051, 177.276, 487.44, 326.336, 487.815, 439.606, 348.496, 602.007, 438.326, 192.483, 327.444, 193.486, 62.814, 21.807, 21.807, 349.765, 308.11, 654.37, 349.988, 349.472, 47.769, 46.945, 41.549, 309.053, 42.191, 295.477, 262.716, 296.565, 40.633, 75.364, 301.068, 298.378, 207.796, 452.952, 331.996]	1.084, 1.098, 1.094, 1.083, 1.094, 1.088, 1.078, 1.087, 1.087, 1.084, 1.099, 1.028, 1.083, 1.096, 1.09, 1.091, 1.094, 1.1, 1.089, 1.1, 1.099, 1.03, 1.095, 1.057, 1.048, 1.035, 1.089, 1.099, 1.075, 1.072, 1.067], [119.672, 119.672, 132.068, 375.623, 239.005, 283.861, 372.218, 82.693, 39.839, 77.27, 168.684, 262.716, 165.485, 481.059, 333.89, 481.456, 438.547, 347.528, 560.144, 437.193, 191.776, 335.254, 192.626, 42.2, 14.733, 14.733, 353.616, 308.087, 661.366, 347.806, 353.895, 45.26, 44.515, 43.937, 308.989, 44.659, 295.545, 262.633, 296.429, 42.799, 78.078, 300.132, 297.571, 207.749, 453.47, 332.667]
7	[-81.379, 150.81, 137.449, 91.559, 138.834, 193.912, 50.951, 30.91, -12.407, 16.76], [1.091, 1.085, 1.086, 1.086, 1.095, 1.095, 1.084, 1.083, 1.092, 1.096, 1.095, 1.082, 1.093, 1.089, 1.079, 1.089, 1.089, 1.087, 1.1, 1.035, 1.086, 1.1, 1.094, 1.093, 1.098, 1.1, 1.09, 1.095, 1.093, 1.043, 1.008, 1.043, 1.041, 1.05, 1.095, 1.1, 1.075, 1.064, 1.077], [118.984, 118.984, 135.082, 371.531, 238.025, 266.695, 368.665, 78.944, 36.29, 77.115, 178.451, 262.088, 173.637, 482.361, 332.529, 482.764, 437.225, 348.491, 563.977, 435.974, 191.283, 333.922, 191.942, 50.984, 14.729, 14.729, 349.735, 304.599, 652.437, 349.985, 349.423, 46.929, 46.129, 42.352, 305.244, 43.022, 294.046, 262.282, 294.305, 38.651, 72.636, 302.094, 299.306, 207.605, 452.984, 331.856]	[-63.251, 78.823, 219.936, 145.754, 91.754, 199.606, 32.107, 41.978, -7.365, -2.428], [1.083, 1.084, 1.084, 1.083, 1.087, 1.086, 1.075, 1.074, 1.081, 1.096, 1.091, 1.08, 1.092, 1.087, 1.077, 1.086, 1.087, 1.085, 1.098, 1.027, 1.082, 1.095, 1.088, 1.09, 1.098, 1.1, 1.089, 1.097, 1.095, 1.046, 1.055, 1.058, 1.047, 1.034, 1.091, 1.089, 1.077, 1.067, 1.065], [122.338, 122.338, 126.917, 371.204, 238.071, 260.795, 368.66, 77.94, 34.974, 78.402, 163.963, 262.935, 162.084, 482.552, 334.145, 482.943, 437.883, 348.835, 558.527, 436.493, 191.465, 335.438, 192.191, 44.795, 13.989, 13.989, 356.43, 307.613, 664.037, 348.732, 356.868, 44.288, 43.571, 44.866, 308.457, 45.623, 295.396, 262.879, 296.094, 40.435, 74.812, 301.176, 298.501, 209.121, 453.29, 332.888]
8	[-80.152, 350.612, 155.979, 89.915, 147.888, 188.027, 54.823, 46.538, -8.233, 1.099], [1.086, 1.083, 1.083, 1.081, 1.093, 1.096, 1.084, 1.081, 1.087, 1.097, 1.096, 1.083, 1.093, 1.087, 1.078, 1.087, 1.088, 1.085, 1.1, 1.036, 1.085, 1.098, 1.092, 1.092, 1.098, 1.1, 1.089, 1.096, 1.095, 1.042, 1.047, 1.047, 1.041, 1.053, 1.093, 1.1, 1.078, 1.066, 1.07], [120.175, 120.175, 130.779, 371.561, 238.182,	[-77.126, 189.015, 174.91, 103.827, 127.487, 187.445, 49.492, 8.462, 5.404, 16.234], [1.087, 1.081, 1.083, 1.084, 1.093, 1.094, 1.082, 1.08, 1.089, 1.097, 1.095, 1.083, 1.094, 1.088, 1.077, 1.086, 1.087, 1.084, 1.098, 1.031, 1.083, 1.096, 1.09, 1.09, 1.094, 1.1, 1.089, 1.1, 1.099, 1.04, 1.029, 1.051, 1.041, 1.045, 1.091, 1.096, 1.066, 1.073, 1.073], [118.825, 118.825, 134.825, 371.606, 238.136, 265.226, 368.572, 79.715, 38.0, 76.785, 176.234, 262.836, 171.735, 479.631, 331.324, 480.016, 439.011, 347.381,

	266.268, 368.972, 77.296, 38.875, 79.707, 188.522, 263.98, 182.787, 492.178, 329.016, 492.587, 443.233, 349.163, 618.663, 441.8, 194.691, 330.233, 196.144, 41.549, 14.953, 14.953, 349.921, 308.667, 654.259, 350.76, 349.569, 48.154, 47.313, 41.189, 309.645, 41.823, 295.653, 263.369, 296.871, 37.072, 69.939, 302.833, 299.952, 207.583, 453.455, 332.0]	570.352, 437.685, 192.122, 332.629, 193.023, 48.696, 14.109, 14.109, 350.723, 307.527, 656.61, 348.363, 350.673, 46.633, 45.844, 42.632, 308.417, 43.311, 295.12, 262.733, 296.03, 42.274, 77.662, 300.713, 298.163, 207.937, 452.962, 332.445]
9	[-97.806, 141.49, 192.276, 107.8, 128.686, 215.864, 32.369, 27.088, 0.563, 33.113], [1.091, 1.081, 1.082, 1.081, 1.093, 1.093, 1.083, 1.081, 1.093, 1.098, 1.095, 1.083, 1.094, 1.087, 1.078, 1.088, 1.088, 1.084, 1.1, 1.034, 1.086, 1.1, 1.092, 1.092, 1.095, 1.1, 1.089, 1.099, 1.098, 1.037, 1.005, 1.055, 1.043, 1.047, 1.098, 1.094, 1.071, 1.07, 1.079], [118.494, 118.494, 140.189, 371.625, 237.802, 273.108, 368.734, 77.427, 41.827, 78.119, 190.294, 264.646, 184.423, 480.766, 330.671, 481.162, 436.972, 347.265, 562.799, 435.708, 191.18, 331.962, 191.791, 57.806, 18.339, 18.339, 351.245, 309.921, 659.196, 348.093, 351.275, 46.811, 46.016, 42.462, 310.962, 43.136, 296.338, 263.974, 297.675, 37.459, 70.624, 302.648, 299.793, 207.082, 453.371, 331.798]	[-73.592, 306.116, 174.522, 98.414, 119.929, 202.537, 36.649, 16.894, 4.914, -0.535], [1.084, 1.081, 1.083, 1.085, 1.094, 1.097, 1.084, 1.082, 1.087, 1.099, 1.097, 1.085, 1.095, 1.089, 1.077, 1.085, 1.086, 1.084, 1.094, 1.027, 1.082, 1.096, 1.088, 1.089, 1.095, 1.1, 1.088, 1.1, 1.099, 1.041, 1.04, 1.053, 1.037, 1.039, 1.092, 1.091, 1.068, 1.072, 1.069], [119.83, 119.83, 130.87, 371.984, 237.889, 264.031, 368.992, 80.661, 35.733, 76.889, 176.715, 262.746, 172.146, 487.68, 330.239, 488.062, 442.967, 347.846, 602.436, 441.55, 194.523, 331.519, 195.943, 38.72, 16.1, 16.1, 350.352, 309.152, 656.597, 349.034, 350.211, 47.409, 46.597, 41.893, 310.169, 42.546, 295.889, 262.589, 297.172, 47.429, 84.075, 298.888, 296.607, 208.916, 452.298, 332.377]
10	[-55.424, -17.928, 432.612, 77.497, 149.093, 212.297, 52.977, 27.083, 2.187, 55.253], [1.091, 1.083, 1.079, 1.07, 1.075, 1.076, 1.066, 1.065, 1.085, 1.1, 1.091, 1.081, 1.094, 1.083, 1.075, 1.086, 1.086, 1.082, 1.096, 1.033, 1.085, 1.1, 1.093, 1.09, 1.096, 1.1, 1.088, 1.099, 1.098, 1.046, 0.956, 1.1, 1.036, 1.05, 1.097, 1.1, 1.072, 1.071, 1.078], [119.172, 119.172, 137.921, 368.365, 238.018, 258.635, 366.624, 82.086, 45.454, 93.354, 165.336, 278.246, 163.057, 478.932, 326.803, 479.325, 434.912, 385.109, 566.346, 433.493, 190.466, 327.633, 190.687, 82.099, 40.778, 40.778, 402.985, 328.115, 729.799, 383.299,	[-99.448, 197.498, 255.89, 116.402, 130.037, 245.414, -26.825, 28.93, 16.648, 8.767], [1.08, 1.075, 1.078, 1.079, 1.089, 1.091, 1.079, 1.077, 1.083, 1.1, 1.096, 1.084, 1.095, 1.086, 1.073, 1.081, 1.082, 1.079, 1.095, 1.029, 1.077, 1.089, 1.077, 1.084, 1.09, 1.097, 1.085, 1.1, 1.1, 1.031, 1.028, 1.069, 1.04, 1.043, 1.091, 1.063, 1.066, 1.075, 1.066], [119.155, 119.155, 132.969, 373.304, 237.77, 273.851, 370.202, 80.643, 37.985, 76.854, 180.981, 265.283, 175.961, 481.651, 331.303, 482.032, 441.278, 348.85, 572.421, 439.84, 193.352, 332.56, 194.545, 41.028, 14.998, 14.998, 356.279, 317.114, 671.975, 348.818, 356.716, 46.254, 45.479, 42.989, 318.489, 43.678, 300.661, 264.498, 302.933, 51.39, 88.302, 297.427, 295.348, 208.887, 453.944, 334.408]

	403.987, 41.028, 40.412, 47.982, 329.753, 48.847, 307.455, 276.376, 310.419, 34.191, 64.009, 304.592, 301.499, 207.624, 452.651, 330.988]	
11	[-102.899, 337.327, 166.861, 121.337, 111.376, 194.449, 51.719, 35.19, -5.41, 29.617], [1.091, 1.082, 1.084, 1.083, 1.093, 1.097, 1.085, 1.083, 1.094, 1.099, 1.097, 1.085, 1.095, 1.089, 1.079, 1.088, 1.088, 1.085, 1.1, 1.031, 1.086, 1.1, 1.093, 1.093, 1.096, 1.1, 1.089, 1.097, 1.096, 1.037, 1.089, 1.051, 1.045, 1.042, 1.095, 1.1, 1.074, 1.067, 1.08], [118.5, 118.5, 139.668, 372.243, 237.934, 275.269, 369.272, 78.324, 39.738, 77.359, 181.312, 263.706, 176.231, 493.31, 326.605, 493.723, 441.409, 348.099, 613.59, 440.053, 193.514, 327.704, 194.748, 55.8, 17.014, 17.014, 349.979, 308.942, 655.586, 349.442, 349.761, 47.656, 46.835, 41.658, 309.939, 42.303, 295.856, 263.206, 297.093, 39.429, 73.664, 301.666, 298.918, 207.194, 453.105, 331.789]	[-29.838, 316.499, 97.24, 103.361, 118.274, 191.227, 64.085, 21.079, 3.16, -17.682], [1.079, 1.084, 1.082, 1.078, 1.086, 1.088, 1.076, 1.074, 1.078, 1.086, 1.086, 1.073, 1.084, 1.081, 1.073, 1.084, 1.085, 1.083, 1.093, 1.026, 1.082, 1.096, 1.09, 1.088, 1.097, 1.1, 1.088, 1.099, 1.099, 1.051, 1.049, 1.026, 1.036, 1.038, 1.091, 1.1, 1.071, 1.072, 1.059], [124.351, 124.351, 123.833, 369.671, 238.044, 253.215, 367.488, 77.22, 36.16, 81.333, 172.956, 261.773, 169.015, 482.913, 332.064, 483.306, 442.599, 352.233, 605.7, 441.067, 194.192, 333.284, 195.515, 35.526, 17.493, 17.493, 350.561, 302.417, 650.179, 354.312, 349.911, 47.623, 46.786, 41.708, 302.798, 42.369, 293.494, 262.109, 293.134, 33.507, 63.35, 305.322, 302.265, 209.559, 452.378, 331.369]
12	[-105.749, 250.714, 134.955, 97.319, 135.1, 192.775, 52.187, 40.588, -7.189, 20.873], [1.09, 1.082, 1.084, 1.085, 1.096, 1.098, 1.086, 1.084, 1.093, 1.098, 1.097, 1.084, 1.094, 1.089, 1.079, 1.088, 1.088, 1.086, 1.1, 1.035, 1.086, 1.1, 1.093, 1.093, 1.097, 1.1, 1.089, 1.097, 1.095, 1.036, 1.059, 1.044, 1.042, 1.049, 1.095, 1.1, 1.075, 1.067, 1.078], [118.565, 118.565, 137.925, 372.871, 238.063, 276.519, 369.824, 79.315, 38.499, 77.046, 188.482, 262.426, 182.697, 480.397, 330.214, 480.768, 439.372, 350.106, 585.115, 438.093, 192.319, 331.543, 193.299, 49.906, 14.486, 14.486, 349.93, 306.334, 652.248, 351.923, 349.461, 48.076, 47.239, 41.262, 307.151, 41.896, 294.577, 262.274, 295.342, 39.387, 73.604, 301.685, 298.935, 207.36, 453.106, 331.855]	[-57.269, 158.667, 168.096, 130.024, 99.8, 202.731, 49.349, 25.677, -2.396, -14.091], [1.081, 1.083, 1.083, 1.081, 1.091, 1.091, 1.08, 1.078, 1.081, 1.095, 1.093, 1.08, 1.091, 1.086, 1.077, 1.087, 1.087, 1.085, 1.098, 1.028, 1.085, 1.1, 1.093, 1.092, 1.096, 1.1, 1.089, 1.098, 1.097, 1.045, 1.007, 1.048, 1.045, 1.037, 1.096, 1.099, 1.072, 1.069, 1.062], [122.416, 122.416, 126.094, 371.233, 237.802, 259.126, 368.539, 77.619, 39.192, 78.923, 180.621, 263.182, 175.56, 480.912, 335.32, 481.308, 439.871, 347.519, 565.128, 438.482, 192.526, 336.725, 193.541, 34.282, 18.68, 18.68, 350.589, 306.59, 655.661, 348.568, 350.498, 46.568, 45.777, 42.699, 307.404, 43.383, 294.664, 262.904, 295.375, 36.158, 68.641, 303.469, 300.555, 207.431, 452.854, 331.522]
13	[-105.38, 271.341, 151.065, 94.81, 137.424, 191.417, 52.527, 36.346, -5.68, 22.499], [1.089, 1.082, 1.084, 1.086, 1.092, 1.095, 1.083, 1.081,	[-68.806, 176.545, 184.245, 126.505, 94.612, 210.728, 41.255, 33.302, -1.453, -3.695], [1.082, 1.082, 1.083, 1.082, 1.089, 1.09, 1.079, 1.077, 1.082, 1.095, 1.093, 1.081, 1.092, 1.086,

	1.091, 1.097, 1.095, 1.083, 1.094, 1.089, 1.079, 1.088, 1.089, 1.086, 1.1, 1.035, 1.086, 1.099, 1.093, 1.093, 1.096, 1.1, 1.089, 1.097, 1.096, 1.036, 1.031, 1.047, 1.042, 1.05, 1.094, 1.1, 1.074, 1.067, 1.076], [118.737, 118.737, 136.706, 373.363, 237.967, 276.36, 370.175, 80.727, 38.225, 76.843, 169.222, 262.225, 165.976, 483.368, 328.701, 483.743, 440.208, 347.682, 590.966, 438.855, 192.754, 329.891, 193.828, 52.652, 15.467, 15.467, 349.866, 305.462, 653.731, 348.953, 349.668, 46.679, 45.888, 42.588, 306.176, 43.265, 294.411, 262.346, 294.845, 39.846, 74.27, 301.475, 298.751, 207.414, 453.094, 331.913]	1.077, 1.086, 1.087, 1.084, 1.095, 1.025, 1.084, 1.098, 1.091, 1.09, 1.096, 1.1, 1.089, 1.098, 1.097, 1.043, 1.009, 1.051, 1.042, 1.033, 1.096, 1.095, 1.074, 1.069, 1.065], [121.404, 121.404, 128.02, 371.542, 237.872, 262.485, 368.83, 78.093, 37.082, 78.138, 171.876, 262.903, 168.055, 479.465, 332.986, 479.853, 439.914, 347.121, 568.015, 438.507, 192.536, 334.295, 193.552, 40.737, 15.018, 15.018, 351.554, 307.229, 657.898, 347.834, 351.625, 45.988, 45.216, 43.249, 308.075, 43.95, 295.057, 262.743, 295.829, 39.302, 73.315, 301.792, 299.065, 208.393, 452.379, 331.602]
14	[-78.008, -20.139, 352.854, 88.683, 147.2, 201.634, 51.665, 23.593, 3.199, 34.357], [1.087, 1.081, 1.081, 1.077, 1.082, 1.081, 1.071, 1.07, 1.084, 1.1, 1.093, 1.082, 1.095, 1.086, 1.078, 1.087, 1.087, 1.084, 1.1, 1.036, 1.086, 1.1, 1.093, 1.092, 1.094, 1.1, 1.089, 1.099, 1.099, 1.04, 1.027, 1.086, 1.04, 1.053, 1.096, 1.1, 1.069, 1.072, 1.073], [119.268, 119.268, 135.028, 370.511, 237.799, 265.534, 367.883, 76.931, 45.105, 81.448, 164.497, 270.11, 162.405, 482.955, 330.794, 483.337, 435.404, 367.937, 566.286, 434.032, 190.548, 331.89, 190.879, 65.195, 24.721, 24.721, 382.386, 318.444, 700.108, 366.355, 383.396, 41.993, 41.35, 47.056, 319.834, 47.886, 301.116, 268.8, 303.284, 36.238, 68.423, 303.248, 300.304, 206.957, 453.308, 331.444]	[-52.885, 137.243, 151.867, 88.73, 146.053, 192.283, 54.78, 36.998, -10.107, 15.295], [1.088, 1.086, 1.084, 1.08, 1.086, 1.086, 1.075, 1.074, 1.085, 1.09, 1.087, 1.076, 1.087, 1.083, 1.076, 1.086, 1.087, 1.085, 1.098, 1.034, 1.084, 1.098, 1.091, 1.091, 1.099, 1.1, 1.089, 1.096, 1.094, 1.049, 0.998, 1.04, 1.039, 1.051, 1.093, 1.099, 1.077, 1.065, 1.072], [120.549, 120.549, 130.693, 369.658, 238.108, 257.969, 367.531, 77.071, 36.074, 81.894, 169.535, 262.392, 166.225, 481.483, 331.599, 481.875, 436.899, 347.041, 562.217, 435.542, 191.106, 332.805, 191.676, 56.294, 17.649, 17.649, 350.214, 304.036, 653.702, 348.071, 350.112, 45.74, 44.97, 43.49, 304.563, 44.206, 294.18, 262.485, 294.109, 33.521, 63.037, 305.187, 302.064, 208.783, 453.182, 331.788]
15	[-85.213, 367.208, 155.427, 115.05, 115.199, 190.831, 51.813, 35.709, -11.583, 9.205], [1.089, 1.084, 1.085, 1.086, 1.094, 1.098, 1.085, 1.083, 1.09, 1.099, 1.098, 1.085, 1.096, 1.09, 1.08, 1.089, 1.089, 1.086, 1.1, 1.032, 1.086, 1.1, 1.093, 1.093, 1.098, 1.1, 1.09, 1.095, 1.094, 1.042, 1.096, 1.05, 1.044, 1.043, 1.094, 1.1, 1.075, 1.064, 1.074], [119.334, 119.334, 133.279, 372.146, 237.936, 268.104, 369.215,	[-82.681, 228.977, 159.036, 95.191, 142.191, 202.532, 36.019, 20.804, -1.047, 13.709], [1.088, 1.082, 1.084, 1.083, 1.095, 1.096, 1.084, 1.082, 1.09, 1.098, 1.096, 1.084, 1.094, 1.088, 1.078, 1.087, 1.088, 1.085, 1.1, 1.035, 1.085, 1.098, 1.091, 1.092, 1.095, 1.1, 1.089, 1.098, 1.097, 1.04, 1.025, 1.049, 1.042, 1.051, 1.094, 1.093, 1.07, 1.07, 1.074], [118.858, 118.858, 134.879, 371.76, 237.862, 267.186, 368.803, 78.713, 38.793, 77.204, 185.133, 263.053, 179.635, 480.171, 330.974, 480.546, 439.741,

	79.189, 36.902, 77.172, 174.067, 262.764, 169.902, 497.343, 327.678, 497.781, 443.293, 348.39, 625.262, 441.884, 194.759, 328.84, 196.226, 45.938, 14.012, 14.012, 349.882, 307.34, 654.236, 349.827, 349.614, 47.507, 46.692, 41.799, 308.234, 42.448, 295.049, 262.567, 295.992, 41.023, 76.034, 301.031, 298.376, 207.439, 452.967, 331.982]	348.345, 579.323, 438.424, 192.52, 332.3, 193.535, 46.807, 13.969, 13.969, 350.007, 307.585, 654.618, 349.736, 349.759, 47.492, 46.674, 41.817, 308.493, 42.469, 295.109, 262.771, 296.099, 39.929, 74.477, 301.596, 298.901, 207.402, 453.431, 332.341]
16	[-79.598, 171.09, 134.184, 90.523, 110.866, 231.934, 43.146, 24.699, 6.845, 30.258], [1.088, 1.081, 1.081, 1.08, 1.086, 1.086, 1.076, 1.074, 1.087, 1.088, 1.087, 1.075, 1.086, 1.082, 1.074, 1.084, 1.085, 1.083, 1.09, 1.021, 1.084, 1.1, 1.092, 1.089, 1.095, 1.1, 1.088, 1.1, 1.1, 1.04, 1.005, 1.035, 1.031, 1.032, 1.1, 1.097, 1.07, 1.073, 1.074], [118.963, 118.963, 135.879, 370.7, 237.876, 266.087, 368.056, 77.551, 37.093, 78.037, 168.042, 261.985, 165.123, 480.158, 330.232, 480.55, 436.933, 347.689, 567.014, 435.578, 191.135, 331.386, 191.707, 60.515, 20.619, 20.619, 349.579, 303.374, 652.077, 349.022, 349.324, 46.157, 45.371, 43.094, 303.82, 43.799, 294.084, 262.392, 293.827, 35.889, 67.631, 303.572, 300.654, 209.281, 452.05, 330.828]	[-81.803, 245.918, 116.497, 121.502, 114.611, 187.969, 54.081, 30.062, 0.956, 1.507], [1.084, 1.081, 1.083, 1.083, 1.092, 1.093, 1.082, 1.08, 1.085, 1.093, 1.092, 1.079, 1.09, 1.086, 1.077, 1.086, 1.087, 1.084, 1.099, 1.031, 1.084, 1.097, 1.091, 1.091, 1.095, 1.1, 1.089, 1.099, 1.098, 1.04, 1.054, 1.036, 1.044, 1.042, 1.092, 1.098, 1.072, 1.071, 1.068], [120.073, 120.073, 130.621, 372.268, 237.795, 266.873, 369.339, 78.892, 38.266, 77.431, 178.494, 261.866, 173.712, 479.83, 332.38, 480.212, 440.268, 350.619, 583.667, 438.894, 192.774, 333.718, 193.849, 40.958, 15.048, 15.048, 350.031, 303.819, 650.953, 352.509, 349.504, 47.595, 46.768, 41.725, 304.384, 42.379, 293.724, 262.086, 293.818, 36.928, 69.781, 303.016, 300.151, 207.901, 453.281, 332.085]
17	[-92.712, 257.229, 156.784, 104.67, 134.851, 231.934, 19.665, 24.568, 2.208, 38.856], [1.092, 1.081, 1.082, 1.079, 1.091, 1.093, 1.082, 1.08, 1.093, 1.095, 1.093, 1.08, 1.091, 1.085, 1.077, 1.087, 1.087, 1.084, 1.1, 1.035, 1.085, 1.1, 1.091, 1.091, 1.095, 1.1, 1.089, 1.099, 1.098, 1.038, 1.07, 1.045, 1.043, 1.049, 1.1, 1.09, 1.07, 1.071, 1.08], [118.512, 118.512, 141.337, 370.837, 237.802, 271.025, 368.114, 76.82, 42.394, 79.365, 189.022, 264.085, 183.3, 482.039, 327.71, 482.417, 438.678, 348.472, 586.846, 437.355, 191.97, 328.833, 192.817, 61.817, 21.291, 21.291, 349.841, 307.704, 654.309, 349.927, 349.553, 47.642, 46.815, 41.677,	[-51.47, 275.55, 79.014, 80.221, 164.163, 158.483, 69.087, 42.96, -7.956, -9.655], [1.083, 1.085, 1.084, 1.082, 1.089, 1.09, 1.078, 1.076, 1.082, 1.087, 1.087, 1.074, 1.085, 1.083, 1.075, 1.085, 1.086, 1.084, 1.098, 1.037, 1.08, 1.092, 1.088, 1.089, 1.099, 1.1, 1.088, 1.096, 1.095, 1.049, 1.057, 1.023, 1.038, 1.056, 1.083, 1.099, 1.078, 1.066, 1.064], [122.943, 122.943, 125.901, 370.597, 238.078, 257.618, 368.312, 77.49, 33.576, 79.33, 169.863, 261.612, 166.492, 479.953, 332.352, 480.338, 440.643, 353.407, 592.225, 439.205, 192.965, 333.625, 194.067, 40.356, 15.114, 15.114, 350.983, 301.494, 650.006, 355.651, 350.24, 47.507, 46.675, 41.814, 301.705, 42.478, 293.839, 262.381, 293.033, 35.096, 66.292, 304.072, 301.095, 210.056, 453.867, 333.228]

	308.594, 42.329, 295.224, 263.514, 296.193, 34.662, 65.686, 304.319, 301.284, 207.142, 453.592, 331.732]	
18	[-81.82, 193.8, 139.652, 107.449, 122.203, 231.934, 16.1, 24.973, -3.197, 7.118], [1.086, 1.082, 1.084, 1.085, 1.092, 1.093, 1.081, 1.08, 1.087, 1.095, 1.093, 1.081, 1.092, 1.088, 1.078, 1.088, 1.088, 1.085, 1.099, 1.032, 1.086, 1.1, 1.091, 1.092, 1.096, 1.1, 1.089, 1.098, 1.096, 1.041, 1.1, 1.042, 1.042, 1.044, 1.1, 1.088, 1.071, 1.069, 1.07], [119.697, 119.697, 131.789, 372.424, 237.843, 266.872, 369.364, 80.165, 37.392, 76.933, 170.394, 262.049, 166.861, 479.666, 332.394, 480.055, 439.107, 347.992, 571.272, 437.766, 192.14, 333.73, 193.058, 45.006, 14.001, 14.001, 349.781, 304.159, 652.608, 349.355, 349.529, 46.52, 45.731, 42.742, 304.746, 43.427, 293.993, 262.379, 294.087, 38.83, 72.844, 302.052, 299.29, 207.653, 452.9, 331.911]	[-54.35, 139.378, 262.181, 101.363, 148.14, 126.954, 80.182, 28.484, -2.872, -3.266], [1.083, 1.084, 1.084, 1.083, 1.086, 1.088, 1.076, 1.074, 1.081, 1.099, 1.094, 1.083, 1.095, 1.088, 1.076, 1.085, 1.086, 1.084, 1.1, 1.036, 1.079, 1.089, 1.086, 1.089, 1.097, 1.1, 1.088, 1.098, 1.097, 1.047, 1.055, 1.069, 1.042, 1.053, 1.076, 1.1, 1.073, 1.069, 1.064], [122.25, 122.25, 126.835, 370.782, 238.003, 258.351, 368.206, 78.287, 34.649, 77.896, 162.626, 264.006, 161.3, 479.341, 331.831, 479.727, 440.156, 352.108, 562.577, 438.704, 192.651, 333.047, 193.684, 43.866, 14.035, 14.035, 361.635, 311.84, 673.473, 351.506, 362.299, 44.061, 43.355, 45.08, 312.967, 45.839, 297.358, 263.634, 298.818, 45.843, 82.008, 299.231, 296.857, 209.532, 454.421, 335.354]
19	[-109.668, 245.585, 176.047, 95.991, 136.456, 196.073, 51.171, 33.346, -0.453, 19.745], [1.088, 1.08, 1.083, 1.084, 1.095, 1.096, 1.085, 1.083, 1.091, 1.1, 1.097, 1.085, 1.095, 1.089, 1.079, 1.088, 1.088, 1.085, 1.1, 1.035, 1.086, 1.1, 1.094, 1.093, 1.095, 1.1, 1.089, 1.098, 1.097, 1.034, 1.1, 1.054, 1.042, 1.05, 1.095, 1.1, 1.072, 1.07, 1.075], [118.562, 118.562, 137.334, 373.607, 237.889, 278.308, 370.384, 79.754, 41.393, 76.957, 184.159, 263.677, 178.751, 481.662, 329.881, 482.031, 440.203, 347.661, 583.644, 438.878, 192.764, 331.165, 193.849, 48.362, 14.202, 14.202, 350.376, 309.132, 656.811, 348.8, 350.26, 47.291, 46.483, 42.004, 310.144, 42.66, 295.919, 263.196, 297.18, 40.137, 74.721, 301.38, 298.676, 206.92, 453.107, 331.732]	[-75.942, 231.243, 204.928, 96.509, 131.905, 199.924, 40.277, 42.456, 2.833, -15.372], [1.079, 1.081, 1.082, 1.081, 1.09, 1.092, 1.08, 1.077, 1.08, 1.098, 1.095, 1.083, 1.093, 1.086, 1.076, 1.085, 1.086, 1.083, 1.096, 1.03, 1.082, 1.095, 1.088, 1.089, 1.096, 1.1, 1.088, 1.099, 1.098, 1.04, 1.036, 1.057, 1.038, 1.044, 1.091, 1.092, 1.075, 1.071, 1.061], [122.465, 122.465, 126.108, 372.33, 238.174, 264.823, 369.582, 78.24, 37.699, 78.349, 178.606, 264.012, 173.798, 481.471, 332.932, 481.85, 442.477, 347.358, 579.872, 441.007, 194.11, 334.256, 195.458, 32.364, 20.211, 20.211, 352.169, 310.807, 661.314, 347.979, 352.297, 46.565, 45.777, 42.698, 311.896, 43.379, 296.781, 263.464, 298.246, 41.994, 76.971, 300.613, 298.039, 208.65, 452.744, 332.304]
20	[-104.169, 482.108, 120.226, 99.046, 140.146, 231.941, 19.208, 28.93, 4.883, 27.317], [1.09, 1.08, 1.081, 1.079, 1.095, 1.1, 1.087, 1.084, 1.094,	[-141.799, 295.947, 55.856, 110.962, 125.56, 191.509, 49.532, 25.938, 7.497, 35.291], [1.091, 1.077, 1.083, 1.088, 1.099, 1.1, 1.089, 1.087, 1.096, 1.094, 1.095, 1.082, 1.091, 1.088,

	1.096, 1.096, 1.083, 1.092, 1.086, 1.077, 1.087, 1.087, 1.084, 1.1, 1.035, 1.085, 1.1, 1.091, 1.091, 1.094, 1.1, 1.089, 1.1, 1.099, 1.034, 1.074, 1.04, 1.042, 1.051, 1.1, 1.089, 1.07, 1.072, 1.078], [118.351, 118.351, 140.549, 372.015, 237.763, 275.848, 369.102, 77.231, 43.752, 78.542, 204.887, 264.41, 198.326, 514.412, 325.32, 515.002, 444.271, 355.182, 677.063, 442.863, 195.502, 326.364, 197.085, 51.183, 14.816, 14.816, 351.825, 308.615, 651.197, 357.619, 351.024, 50.017, 49.111, 39.428, 309.607, 40.007, 295.621, 263.658, 296.955, 35.921, 68.099, 303.545, 300.603, 206.851, 453.565, 331.754]	1.078, 1.088, 1.088, 1.085, 1.1, 1.033, 1.085, 1.098, 1.092, 1.092, 1.092, 1.1, 1.089, 1.1, 1.1, 1.025, 1.098, 1.025, 1.044, 1.046, 1.093, 1.098, 1.067, 1.073, 1.081], [118.584, 118.584, 145.107, 376.829, 237.774, 294.696, 373.051, 87.707, 40.179, 78.627, 185.067, 261.465, 179.567, 480.286, 330.059, 480.659, 438.837, 360.486, 599.124, 437.606, 192.071, 331.421, 192.984, 52.021, 15.078, 15.078, 354.784, 301.827, 650.485, 363.359, 353.702, 49.229, 48.349, 40.174, 302.165, 40.777, 293.315, 262.284, 292.878, 40.495, 75.23, 301.295, 298.625, 207.076, 453.371, 332.126]

Annex B – Data sheets for IEEE test power systems

B.1 – IEEE 14 bus system

B.1.1 – BRANCH DATA

From Bus	To Bus	Resistance [p.u]	Reactance [p.u]	Charging susceptance [p.u]	MVA rating
1	2	0,01938	0,05917	0,11834	120
1	5	0,05403	0,22304	0,44608	65
2	3	0,04699	0,19797	0,39594	36
2	4	0,05811	0,17632	0,35264	65
2	5	0,05695	0,17388	0,34776	50
3	4	0,06701	0,17103	0,34206	65
4	5	0,01335	0,04211	0,08422	45
4	7	0	0,20912	0	55
4	9	0	0,55618	0	32
5	6	0	0,25202	0	45
6	11	0,09498	0,1989	0	18
6	12	0,12291	0,25581	0	32
6	13	0,06615	0,13027	0	32
7	8	0	0,17615	0	32
7	9	0	0,11001	0	32
9	10	0,03181	0,0845	0	32
9	14	0,12711	0,27038	0	32
10	11	0,08205	0,19207	0	12
12	13	0,22092	0,19988	0	12
13	14	0,17093	0,34802	0	12

B.1.2 – TRANSFORMER TAP SETTING DATA

From bus	To bus	Tap setting value	Step
4	**7**	0,978	0,0125
4	**9**	0,969	0,0125
5	**6**	0,932	0,0125

B.1.3 – BUS DATA

Bus number	Magnitude [p.u]	Load angle [º]	Generator real power [MW]	Generator reactive power [Mvar]	Load real power [MW]	Load reactive power [Mvar]	Minimum reactive power [Mvar]	Maximum reactive power [Mvar]
1	1,06	0	232,39	-16,9	0	0	0	10
2	1,045	0	40	41,9	21,7	12,7	-48	64
3	1,01	0	0	23,6	94,2	19,1	-30	40
4	1	0	-	-	47,8	-3,9	-	-
5	1	0	-	-	7,6	1,6	-	-
6	1,07	0	0	12,2	11,2	7,5	-18	24
7	1	0	-	-	0	0	-	-
8	1,09	0	0	17,3	0	0	-18	24
9	1	0	-	-	29,5	16,6	-	-
10	1	0	-	-	9	5,8	-	-
11	1	0	-	-	3,5	1,8	-	-
12	1	0	-	-	6,1	1,6	-	-
13	1	0	-	-	13,8	5,8	-	-
14	1	0	-	-	14,9	5	-	-

B.1.4 – SHUNT CAPACITOR DATA

Bus number	Susceptance [Mvar]	Step [Mvar]
9	18	6

B.2 – IEEE 39 bus system

B.2.1 – BRANCH DATA

From Bus	To Bus	Resistance [p.u]	Reactance [p.u]	Charging susceptance [p.u]	MVA rating
1	2	0,0035	0,0411	0,6987	-
1	39	0,001	0,025	0,75	-
2	3	0,0013	0,0151	0,2572	-
2	25	0,007	0,0086	0,146	-
3	4	0,0013	0,0213	0,2214	-

3	18	0,0011	0,0133	0,2138	-
4	5	0,0008	0,0128	0,1342	-
4	14	0,0008	0,0129	0,1382	-
5	6	0,0002	0,0026	0,0434	-
5	8	0,0008	0,0112	0,1476	-
6	7	0,0006	0,0092	0,113	-
6	11	0,0007	0,0082	0,1389	-
7	8	0,0004	0,0046	0,078	-
8	9	0,0023	0,0363	0,3804	-
9	39	0,001	0,025	1,2	-
10	11	0,0004	0,0043	0,0729	-
10	13	0,0004	0,0043	0,0729	-
13	14	0,0009	0,0101	0,1723	-
14	15	0,0018	0,0217	0,366	-
15	16	0,0009	0,0094	0,171	-
16	17	0,0007	0,0089	0,1342	-
16	19	0,0016	0,0195	0,304	-
16	21	0,0008	0,0135	0,2548	-
16	24	0,0003	0,0059	0,068	-
17	18	0,0007	0,0082	0,1319	-
17	27	0,0013	0,0173	0,3216	-
21	22	0,0008	0,014	0,2565	-
22	23	0,0006	0,0096	0,1846	-
23	24	0,0022	0,035	0,361	-
25	26	0,0032	0,0323	0,513	-
26	27	0,0014	0,0147	0,2396	-
26	28	0,0043	0,0474	0,7802	-
26	29	0,0057	0,0625	1,029	-
28	29	0,0014	0,0151	0,249	-
12	11	0,0016	0,0435	0	-
12	13	0,0016	0,0435	0	-
6	31	0	0,025	0	-
10	32	0	0,02	0	-
19	33	0,0007	0,0142	0	-
20	34	0,0009	0,018	0	-
22	35	0	0,0143	0	-
23	36	0,0005	0,0272	0	-
25	37	0,0006	0,0232	0	-
2	30	0	0,0181	0	-
29	38	0,0008	0,0156	0	-
19	20	0,0007	0,0138	0	-

B.2.2 – TRANSFORMER TAP SETTING DATA

From bus	To bus	Tap setting value	Step
6	**31**	0,975	0,0125

B.2.3 – BUS DATA

Bus Number	Magnitude [p.u]	Load angle [º]	Generator real power [MW]	Generator reactive power [Mvar]	Load real power [MW]	Load reactive power [Mvar]	Minimum reactive power [Mvar]	Maximum reactive power [Mvar]
1	1,0489	-10,21	250	125,3736	0	0	-	-
2	1,0524	-7,68	-	-	0	0	-	-
3	1,0387	-10,52	650	164,3574	322	2,4	-	-
4	1,0217	-11,32	632	97,5603	500	184	-	-
5	1,0208	-10,17	508	161,7146	0	0	-	-
6	1,0182	-9,47	650	198,3432	0	0	-	-
7	1,0088	-11,62	560	93,1968	233,8	84	-	-
8	1,0083	-12,12	540	-10,6838	522	176	-	-
9	1,0333	-11,93	830	15,5849	0	0	-	-
10	1,0259	-7,12	1000	60,9293	0	0	-	-
11	1,0221	-7,92	-	-	0	0	-	-
12	1,01	-7,92	-	-	7,5	88	-	-
13	1,0241	-7,82	-	-	0	0	-	-
14	1,0239	-9,47	-	-	0	0	-	-
15	1,0228	-9,85	-	-	320	153	-	-
16	1,037	-8,44	-	-	329,4	32,3	-	-
17	1,0393	-9,43	-	-	0	0	-	-
18	1,0378	-10,27	-	-	158	30	-	-
19	1,0517	-3,83	-	-	0	0	-	-
20	0,9919	-5,24	-	-	680	103	-	-
21	1,0354	-6,05	-	-	274	115	-	-
22	1,0518	-1,62	-	-	0	0	-	-
23	1,0468	-1,82	-	-	247,5	84,6	-	-
24	1,042	-8,32	-	-	308,6	-92,2	-	-
25	1,0601	-6,29	-	-	224	47,2	-	-
26	1,0553	-7,56	-	-	139	17	-	-
27	1,0421	-9,56	-	-	281	75,5	-	-
28	1,0518	-4,05	-	-	206	27,6	-	-
29	1,0511	-1,3	-	-	-	-	-	-
30	1,0475	-5,27	-	-	-	-	-500	800
31	0,982	-1,59	-	-	283,5	26,9	-500	800

32	0,9831	0,81	-	-			-500	800
33	0,9972	1,38	-	-	-	-	-500	800
34	1,0123	-0,05	-	-	-	-	-300	800
35	1,0493	3,33	-	-	-	-	-500	800
36	1,0635	6,02	-	-	-	-	-500	800
37	1,0278	0,48	-	-	-	-	-500	800
38	1,0265	5,76	-	-	-	-	-500	800
39	1,03	-11,73	-	-	9,2	4,6	-1000	1500

B.2.4 – SHUNT CAPACITOR DATA

Bus number	Susceptance [Mvar]	Step [Mvar]
4	100	25
5	200	25

Annex C – Additional Results

Table 17 – Results of the global PSO with different numbers of particles for the IEEE 14 bus system

Optimized power losses [MW]				
PSO				
Run no.	**60 particles**	**90 particles**	**120 particles**	**150 particles**
1	12,39536	12,30729	12,3619	12,29695
2	12,39695	12,37376	12,3777	12,28295
3	12,47182	12,28226	12,28226	12,27964
4	12,41634	12,42588	12,28078	12,28078
5	12,30905	12,35171	12,27964	12,42356
6	12,39206	12,37688	12,33933	12,39092
7	12,33905	12,41119	12,33905	12,33905
8	12,32934	12,32035	12,34188	12,27964
9	12,46029	12,32026	12,37376	12,27964
10	12,43708	12,3958	12,33933	12,3534
11	12,33905	12,37228	12,36801	12,37657
12	12,39792	12,27964	12,28521	12,28078
13	12,30229	12,32011	12,30751	12,27964
14	12,34205	12,36849	12,30242	12,41255
15	12,34785	12,32033	12,37251	12,27964
16	12,36801	12,33904	12,34785	12,28078
17	12,44407	12,28847	12,40995	12,27964
18	12,27964	12,29695	12,32026	12,27964
19	12,37557	12,37657	12,32934	12,27964
20	12,32934	12,41321	12,29317	12,37883
21	12,34785	12,33968	12,28078	12,36849
22	12,40426	12,32525	12,41255	12,33905
23	12,41339	12,30659	12,28072	12,27964
24	12,40995	12,36801	12,29151	12,3231
25	12,32538	12,39237	12,30489	12,33905
26	12,39237	12,33904	12,30481	12,37883
27	12,38606	12,41972	12,37376	12,39237
28	12,45302	12,34785	12,32306	12,34631
29	12,27964	12,30736	12,29695	12,27964
30	12,35502	12,30751	12,3254	12,41484
Performance indicators				
	60 particles	**90 particles**	**120 particles**	**150 particles**
Avg. acc. [MW]	12,375	12,346	12,328	12,327

Std. Dev.	0,052	0,043	0,039	0,051
Best acc [MW]	12,280	12,280	12,280	12,280
Success %	7	7	17	17

Table 18 – Results of the global PSO with different numbers of particles for the IEEE 39 bus system

Optimized power losses [MW]				
PSO				
Run no.	**90 particles**	**120 particles**	**150 particles**	**180 particles**
1	37,9964	38,01718	38,02657	38,02799
2	38,06409	38,53315	37,97356	37,97873
3	38,31963	37,97009	37,96733	37,99041
4	38,07994	38,00389	38,52899	38,38721
5	38,03488	38,03267	38,00616	38,01645
6	37,99274	38,09457	37,99953	38,00021
7	38,08119	37,95535	38,02385	38,11519
8	38,02762	38,09384	38,02	38,06237
9	38,08977	38,01808	37,99459	38,02095
10	38,05508	38,12134	38,02947	38,04851
11	38,01751	37,98215	37,99103	37,98393
12	38,01685	37,97688	37,97643	38,0564
13	38,07665	38,1972	38,00234	37,9684
14	38,02191	38,04568	38,06039	38,05414
15	38,14493	38,03654	38,09982	37,9651
16	38,03887	38,04704	37,99247	38,09859
17	37,97485	37,97478	38,01222	37,99246
18	38,00144	38,11626	38,0637	37,97131
19	38,12218	37,9692	38,0456	38,07233
20	38,11272	38,13555	38,01199	38,01374
21	38,1605	38,02879	38,01735	37,95528
22	37,96153	38,53021	38,04045	38,02566
23	38,10144	38,01482	38,05556	38,016
24	38,08176	37,97407	38,12088	38,09466
25	38,0201	38,10168	37,99832	37,96381
26	37,98905	38,04677	38,05391	38,00809
27	38,04308	37,98416	38,08878	37,9846
28	38,31466	38,1143	37,98964	37,98319
29	38,2102	38,02568	38,05573	37,98468
30	38,11574	37,98708	37,998	38,01437

	90 particles	120 particles	150 particles	180 particles
Avg. acc. [MW]	38,076	38,071	38,041	38,028
Std. Dev.	0,087	0,139	0,099	0,080
Best acc [MW]	37,962	37,955	37,967	37,955
Success %	3	3	7	7

Annex D – SO-PSO Results

Table 19 - Detailed results of 10 runs of the SO-PSO and FDR-SO-PSO for the IEEE 14 bus

Optimized power losses [MW]			
SO-PSO run no.	**Run no.**	**PSO**	**FDR-PSO**
21	**1**	12,45717	12,28151
	2	12,3826	12,27964
	3	12,33905	12,27964
	4	12,33905	12,27964
	5	12,33905	12,27964
	6	12,28078	12,27964
	7	12,27964	12,27964
	8	12,27964	12,27964
	9	12,27964	12,27964
	10	12,27964	12,27964
22	**1**	12,32195	12,28844
	2	12,32033	12,28295
	3	12,32033	12,28295
	4	12,32033	12,28069
	5	12,32033	12,28069
	6	12,32033	12,28069
	7	12,32033	12,28069
	8	12,32033	12,28069
	9	12,32033	12,27964
	10	12,32033	12,27964
23	**1**	12,4603	12,30651
	2	12,46027	12,29115
	3	12,46025	12,2808
	4	12,45821	12,27964
	5	12,45821	12,27964
	6	12,41288	12,27964
	7	12,40116	12,27964
	8	12,39613	12,27964
	9	12,39613	12,27964
	10	12,39613	12,27964
	1	12,27964	12,29656
	2	12,27964	12,28295

24	**3**	12,27964	12,28074
	4	12,27964	12,27964
	5	12,27964	12,27964
	6	12,27964	12,27964
	7	12,27964	12,27964
	8	12,27964	12,27964
	9	12,27964	12,27964
	10	12,27964	12,27964
25	**1**	12,28069	12,27964
	2	12,28069	12,27964
	3	12,28069	12,27964
	4	12,28069	12,27964
	5	12,28069	12,27964
	6	12,28069	12,27964
	7	12,28069	12,27964
	8	12,27964	12,27964
	9	12,27964	12,27964
	10	12,27964	12,27964
26	**1**	12,42678	12,28072
	2	12,39237	12,28072
	3	12,39237	12,28072
	4	12,39237	12,28072
	5	12,39237	12,28072
	6	12,39237	12,28072
	7	12,39237	12,27964
	8	12,39237	12,27964
	9	12,39237	12,27964
	10	12,39237	12,27964
27	**1**	12,28913	12,30275
	2	12,27964	12,28814
	3	12,27964	12,28295
	4	12,27964	12,28295
	5	12,27964	12,28073
	6	12,27964	12,28072
	7	12,27964	12,28072
	8	12,27964	12,28072
	9	12,27964	12,28072
	10	12,27964	12,27964
	1	12,37376	12,29665
	2	12,32033	12,28864
	3	12,32033	12,28309
	4	12,32033	12,28069

28	**5**	12,32033	12,27964
	6	12,32033	12,27964
	7	12,32033	12,27964
	8	12,32033	12,27964
	9	12,32033	12,27964
	10	12,28794	12,27964
29	**1**	12,39487	12,29141
	2	12,30729	12,28158
	3	12,30729	12,28073
	4	12,30729	12,27964
	5	12,30729	12,27964
	6	12,30729	12,27964
	7	12,30729	12,27964
	8	12,30729	12,27964
	9	12,30729	12,27964
	10	12,30729	12,27964
30	**1**	12,34179	12,30836
	2	12,33905	12,29291
	3	12,27964	12,28296
	4	12,27964	12,28069
	5	12,27964	12,27964
	6	12,27964	12,27964
	7	12,27964	12,27964
	8	12,27964	12,27964
	9	12,27964	12,27964
	10	12,27964	12,27964

Table 20 - Detailed results of 10 runs of the SO-PSO and FDR-SO-PSO for the IEEE 39 bus

Optimized power losses [MW]			
SO-PSO run no.	**Run no.**	**PSO**	**FDR-PSO**
21	**1**	38,11416	37,97463
	2	38,05827	37,96727
	3	38,05337	37,96419
	4	38,0237	37,96395
	5	38,02309	37,95827
	6	38,02288	37,95723
	7	38,02284	37,95706
	8	38,02261	37,95636
	9	38,02257	37,95622
	10	38,02224	37,95621

22	**1**	38,04034	38,09165
	2	37,99008	37,99447
	3	37,98954	37,97299
	4	37,98954	37,96169
	5	37,98941	37,95855
	6	37,98941	37,95498
	7	37,98941	37,95425
	8	37,98941	37,95354
	9	37,98939	37,95341
	10	37,98938	37,95263
23	**1**	38,16501	38,13258
	2	38,13344	37,98611
	3	38,107	37,97239
	4	38,08335	37,96997
	5	38,07139	37,96901
	6	38,07054	37,96852
	7	38,06897	37,96726
	8	38,06731	37,96631
	9	38,06731	37,96611
	10	38,05857	37,96606
24	**1**	37,98787	38,38863
	2	37,98776	38,14513
	3	37,98776	37,98034
	4	37,98764	37,95499
	5	37,9872	37,95298
	6	37,98695	37,94978
	7	37,98693	37,9497
	8	37,98693	37,94968
	9	37,98693	37,94912
	10	37,98654	37,94888
25	**1**	38,04511	38,04323
	2	38,04295	37,99467
	3	38,03405	37,97084
	4	38,03405	37,9665
	5	38,02441	37,96429
	6	38,0205	37,96019
	7	38,02039	37,95806
	8	38,02035	37,95732
	9	38,02021	37,9572
	10	38,01916	37,95683
	1	38,0434	38,17176

26	**2**	38,0434	38,01318
	3	38,0434	37,97164
	4	38,0391	37,96648
	5	38,0391	37,9545
	6	38,03825	37,95335
	7	38,03825	37,95271
	8	38,03771	37,95141
	9	38,03718	37,95036
	10	38,03564	37,95024
27	**1**	38,16886	38,02254
	2	38,14399	37,97593
	3	38,13737	37,95721
	4	38,13719	37,95377
	5	38,13474	37,95339
	6	38,13299	37,95293
	7	38,13139	37,95282
	8	38,05797	37,95282
	9	38,0549	37,95038
	10	38,05465	37,9503
28	**1**	38,12529	38,01098
	2	38,10313	37,99563
	3	38,08825	37,99388
	4	38,07874	37,99281
	5	38,07874	37,99258
	6	38,07219	37,96671
	7	38,05947	37,96599
	8	38,05925	37,96579
	9	38,05844	37,96544
	10	38,0583	37,94974
29	**1**	38,02118	38,10162
	2	38,00159	37,98643
	3	38,00044	37,9791
	4	38,00044	37,97204
	5	38,00044	37,97141
	6	38,00044	37,97063
	7	38,00044	37,96635
	8	38,00044	37,96459
	9	38,00044	37,96341
	10	38,00042	37,96297
	1	38,09904	38,40009
	2	38,05377	38,03552
	3	38,01017	37,96978

30	**4**	37,99569	37,96236
	5	37,99506	37,95837
	6	37,99027	37,95629
	7	37,98502	37,95588
	8	37,98502	37,9554
	9	37,98391	37,95508
	10	37,98389	37,95461

Annex E – Optimal solutions and power losses with ON/OFF shunt capacitor

E.1 – Power losses

Table 21 - Optimized power losses for IEEE 14 bus considering ON/OFF shunt capacitor battery

Optimized power losses [MW]		
Run no.	**PSO (IEEE 14)**	**PSO (IEEE 39)**
1	12,392	38,029
2	12,373	38,018
3	12,426	38,177
4	12,395	38,066
5	12,341	38,037
6	12,307	38,058
7	12,320	38,127
8	12,339	37,990
9	12,285	38,113
10	12,281	38,139
11	12,280	38,051
12	12,285	38,045
13	12,375	37,980
14	12,377	38,380
15	12,295	38,038
16	12,341	38,102
17	12,453	38,081
18	12,376	38,082
19	12,484	37,982
20	12,310	38,082

E.2 – Optimal Solutions

Table 22 - Optimal solutions for IEEE 14 bus considering ON/OFF shunt capacitor battery

Run no.	PSO (IEEE 14)	PSO (IEEE 39)
1	[[0], [1.0375, 0.9, 1.0], [1.1, 1.086, 1.1, 1.095, 1.1]]	[[100, 0], [0.925], [1.031, 1.062, 1.052,

		1.042, 1.05, 1.099, 1.09, 1.068, 1.072, 1.081]]
2	[[18], [1.0125, 0.9, 0.975], [1.1, 1.1, 1.055, 1.1, 1.1]]	[[100, 0], [0.9], [1.036, 1.031, 1.052, 1.041, 1.052, 1.099, 1.092, 1.069, 1.071, 1.077]]
3	[[18], [1.1, 0.9, 1.0], [1.1, 1.1, 1.058, 1.1, 1.1]]	[[100, 0], [0.9125], [1.043, 1.048, 1.035, 1.025, 1.028, 1.1, 1.099, 1.072, 1.073, 1.077]]
4	[[0], [0.975, 1.0875, 1.0375], [1.1, 1.086, 1.057, 1.1, 1.1]]	[[100, 200], [0.9125], [1.036, 1.021, 1.029, 1.043, 1.045, 1.1, 1.088, 1.074, 1.068, 1.078]]
5	[[18], [1.025, 0.9, 0.9875], [1.1, 1.086, 1.1, 1.1, 1.092]]	[[100, 200], [0.9875], [1.036, 1.066, 1.061, 1.045, 1.043, 1.094, 1.1, 1.068, 1.071, 1.074]]
6	[[18], [1.1, 0.9, 1.0], [1.1, 1.087, 1.058, 1.1, 1.1]]	[[0, 200], [0.9], [1.041, 1.009, 1.052, 1.044, 1.045, 1.096, 1.1, 1.072, 1.069, 1.075]]
7	[[18], [0.9625, 1.1, 1.0125], [1.1, 1.086, 1.056, 1.097, 1.1]]	[[0, 200], [0.9], [1.036, 1.016, 1.048, 1.041, 1.045, 1.1, 1.089, 1.075, 1.068, 1.085]]
8	[[18], [1.0125, 0.9, 0.9875], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[100, 200], [0.975], [1.033, 1.087, 1.041, 1.043, 1.046, 1.096, 1.099, 1.067, 1.072, 1.071]]
9	[[18], [1.0, 0.9375, 0.9875], [1.1, 1.086, 1.057, 1.1, 1.1]]	[[0, 200], [0.975], [1.05, 1.095, 1.045, 1.042, 1.044, 1.096, 1.1, 1.07, 1.067, 1.072]]
10	[[18], [1.0125, 0.9125, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[0, 200], [0.975], [1.044, 1.087, 1.037, 1.041, 1.049, 1.1, 1.092, 1.073, 1.07, 1.073]]
11	[[18], [1.025, 0.9, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[0, 200], [0.975], [1.038, 1.091, 1.052, 1.043, 1.049, 1.096, 1.1, 1.071, 1.07, 1.078]]
12	[[18], [0.9875, 0.9375, 0.975], [1.1, 1.085, 1.055, 1.1, 1.1]]	[[100, 200], [1.0], [1.038, 1.077, 1.056, 1.043, 1.049, 1.091, 1.1, 1.072, 1.069, 1.082]]
13	[[18], [0.95, 1.1, 1.0125], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[100, 200], [0.9875], [1.033, 1.1, 1.05, 1.04,

		1.051, 1.095, 1.099, 1.071, 1.069, 1.073]]
14	[[18], [1.1, 0.9, 1.0125], [1.1, 1.086, 1.1, 1.1, 1.078]]	[[100, 0], [0.925], [1.042, 1.007, 1.087, 1.044, 1.047, 1.1, 1.088, 1.07, 1.073, 1.061]]
15	[[18], [1.075, 0.9, 1.0], [1.1, 1.087, 1.058, 1.098, 1.1]]	[[100, 200], [0.975], [1.04, 1.077, 1.032, 1.041, 1.038, 1.097, 1.1, 1.071, 1.07, 1.075]]
16	[[18], [0.9875, 0.9375, 0.975], [1.1, 1.085, 1.1, 1.1, 1.1]]	[[100, 0], [0.925], [1.04, 1.064, 1.016, 1.046, 1.045, 1.097, 1.1, 1.073, 1.072, 1.071]]
17	[[0], [1.1, 0.9, 1.0375], [1.1, 1.086, 1.1, 1.1, 1.1]]	[[100, 200], [1.0125], [1.033, 1.099, 1.055, 1.035, 1.045, 1.1, 1.091, 1.069, 1.073, 1.08]]
18	[[18], [1.1, 0.9, 1.0], [1.1, 1.086, 1.1, 1.09, 1.082]]	[[0, 200], [1.0], [1.038, 1.1, 1.059, 1.043, 1.049, 1.096, 1.1, 1.071, 1.071, 1.08]]
19	[[18], [0.925, 0.925, 0.975], [1.1, 1.1, 1.055, 1.1, 0.9]]	[[100, 200], [0.9], [1.034, 1.003, 1.041, 1.041, 1.045, 1.095, 1.1, 1.071, 1.069, 1.078]]
20	[[18], [1.1, 0.9, 1.0125], [1.1, 1.087, 1.058, 1.096, 1.1]]	[[100, 200], [0.9875], [1.042, 1.067, 1.063, 1.042, 1.048, 1.099, 1.089, 1.071, 1.067, 1.075]]

Annex F – Implemented Algorithm in Python

```
from __future__ import division
import random
import math
import pss
import time
import matplotlib.pyplot as plt


# evaluate constraint violation for state variables
def constraint_violation(power_system):
    ineq_result = []
    ineq_is_viol = [[False for i in range(0, pss.n_gen)], [False for i in
range(0, pss.n_buses)],
                    [False for i in range(0, pss.n_branches)]]
    volt_lim = (0.9, 1.1)
    S_lim = (-1000, 1000)
    y_lim = [pss.Qgen_lim, [volt_lim for i in range(0, pss.n_buses)], [S_lim for
i in range(0, pss.n_branches)]]
    state_var = power_system.state_var
    num_statevar = 3
    toler = 0.00001

    viol_g = [[0 for i in range(0, pss.n_gen)], [0 for i in range(0,
pss.n_buses)], [0 for i in range(0, pss.n_branches)]]

    #Calculate g(y):
    for i in range(0, num_statevar):
        for j in range(0, len(state_var[i])):
            if state_var[i][j] < y_lim[i][j][0] - toler:
                viol_g[i][j] = -state_var[i][j] + y_lim[i][j][0]
            if state_var[i][j] > y_lim[i][j][1] + toler:
                viol_g[i][j] = state_var[i][j] - y_lim[i][j][1]

    #Calculate overall constraint violation:
    v = 0
    num = 0
    denom = 0
    for i in range(0, num_statevar):
        if max(viol_g[i]) > 0:
            for j in range(0, len(state_var[i])):
                num += viol_g[i][j]/max(viol_g[i])
            denom += 1/max(viol_g[i])

    if denom > 0:
        v = num / denom

    return v #overall constraint violation

class Particle:
    def __init__(self, x0):

        self.position_i = [i[:] for i in controlvect]  # particle position
        self.velocity_i = [i[:] for i in controlvect] # particle velocity
        self.pos_best_i = [i[:] for i in controlvect]  # best individual position
        self.func_best_i = -1  # best fitness individual
```

```
        self.func_i = -1  # fitness individual
        self.constr_best_i = [-1, -1, -1] # least constraint violation individual
        self.constr_i = [-1,- 1, -1] # constraint violation individual
        self.neigh_constr_i = [-1,- 1, -1] # constraint violation for neighbor
        self.neigh_constr_best_i = [-1,- 1, -1] # least constraint violation for
neighbor
        self.fdr_i = []  # fitness distance ratio for each neighbourhood particle
        self.fdr_best_i = []
        self.func_near = [-1, -1]
        self.func_near_best = -1
        self.pos_near_best = [i[:] for i in controlvect0] # best local position
        self.better_constr = False
        self.neigh_better_constr = False
        self.pos_best_i_arr = [i[:] for i in controlvect] * 2

        for i in range(0, num_controlvect):
            for j in range(0, len(controlvect[i])):
                self.velocity_i[i][j] = random.uniform(-1, 1)
                self.position_i[i][j] = x0[i][j]

    # evaluate current fitness
    def evaluate(self, power_system):
        power_system.solve_power_flow(self.position_i) # solve power flow
        self.func_i = power_system.Ploss # get active power losses

        self.constr_i = constraint_violation(power_system)

        self.better_constr = False

        if self.constr_i <= self.constr_best_i or self.constr_best_i == -1:
            self.better_constr = True

        # check if the current position is an individual best with the principle
of constrained non-domination
        if ((self.func_i < self.func_best_i and self.better_constr) or
self.better_constr) or self.func_best_i == -1:
            self.pos_best_i = [x[:] for x in self.position_i]
            self.func_best_i = self.func_i
            self.constr_best_i = self.constr_i


    # find local best based on fitness-distance ratio (FDR)
    def get_local_best(self, power_system, neighbours):

        neigh = [[] for i in range(0, len(neighbours))]
        position_i = self.position_i[0] + self.position_i[1] + self.position_i[2]

        for i in range(0, len(neighbours)):
            neigh[i] = neighbours[i][0] + neighbours[i][1] + neighbours[i][2]

        for k in range(0, len(neighbours)):
            if neigh[k] != position_i:

                power_system.solve_power_flow(neighbours[k])
                neigh_func = power_system.Ploss

                fit_dif = -(neigh_func - self.func_i) # fitness difference

                sum = 0.0
                for n in range(0, len(neigh[k])):
                    sum = sum + (neigh[k][n]-position_i[n])**2
```

```
                distance = abs(math.sqrt(sum)) # distance

                if abs(fit_dif - distance) > 0.000000000001:
                    self.fdr_i = fit_dif / distance #fi

            # nBest is selected such that the FDR is maximized:
            if self.fdr_i > self.fdr_best_i or self.fdr_best_i == []:
                self.pos_near_best = [x[:] for x in neighbours[k]]
                self.fdr_best_i = self.fdr_i

    # update new particle velocity
    def update_velocity(self, pos_best_g, bounds, k, maxiter):
        w = 0.9 - ((0.9-0.4)/maxiter)*k
        c1 = 1  # cognitive constant
        c2 = 1  # social constant, BETTER RESULTS with c2=1 than c2=2
        c3 = 1  # local constant
        phi = c1 + c2 + c3
        X = 2 / abs(2 - phi - math.sqrt(abs(phi ** 2 - 4 * phi)))
        vmax = [i[:] for i in controlvect]
        vmin = [i[:] for i in controlvect]

        for i in range(0, num_controlvect):
            for j in range(0, len(controlvect[i])):
                vmax[i][j] = (bounds[i][j][1]-bounds[i][j][0])*0.5
                vmin[i][j] = (bounds[i][j][1] - bounds[i][j][0]) * 0.001
                r1 = random.random()
                r2 = random.random()
                r3 = random.random()

                vel_cognitive = c1 * r1 * (self.pos_best_i[i][j] -
self.position_i[i][j])
                vel_social = c2 * r2 * (pos_best_g[i][j] - self.position_i[i][j])

                if is_local_pso:
                    vel_local = c3 * r3 * (self.pos_near_best[i][j] -
self.position_i[i][j])
                else:
                    vel_local = 0

                self.velocity_i[i][j] = X*(w * self.velocity_i[i][j] +
vel_cognitive + vel_social + vel_local)

                #if self.velocity_i[i][j] > vmax[i][j]: #better results without
this
                #    self.velocity_i[i][j] = vmax[i][j]
                #if self.velocity_i[i][j] < vmin[i][j]:
                #     self.velocity_i[i][j] = vmin[i][j]

    # update the particle position based off new velocity updates
    def update_position(self, bounds):

        for i in range(0, num_controlvect):
            for j in range(0, len(controlvect[i])):

                self.position_i[i][j] = self.position_i[i][j] +
self.velocity_i[i][j]

                # adjust maximum position if necessary
                if self.position_i[i][j] > bounds[i][j][1]:
                    self.position_i[i][j] = bounds[i][j][1]
```

```
                # adjust minimum position if necessary
                if self.position_i[i][j] < bounds[i][j][0]:
                    self.position_i[i][j] = bounds[i][j][0]


        self.pos_near_best = [[0 for i in range(0, pss.n_cap)], [0 for i in
range(0, pss.n_transf)], [0 for i in range(0, pss.n_gen)]]
        self.fdr_best_i = []
        self.func_near_best = []

    #convert Qc and T to discrete values according to some article
    def discretize(self, Qcdis, Tdis):
        hcube = [[[0 for j in range(0, 2)] for i in range(0, pss.n_cap)],
                 [[0 for j in range(0, 2)] for i in range(0, pss.n_transf)]] #
local hypercube
        x_low = [[0 for i in range(0, pss.n_cap)], [0 for i in range(0,
pss.n_transf)]]  # upper bound of discrete var
        x_up = [[0 for i in range(0, pss.n_cap)], [0 for i in range(0,
pss.n_transf)]]  # lower bound of discrete var

        for i in range(0, pss.n_cap):
            for j in range(1, len(Qcdis[i])):
                if Qcdis[i][j-1] <= self.position_i[0][i] <= Qcdis[i][j]:
                    x_low[0][i] = Qcdis[i][j-1]
                    x_up[0][i] = Qcdis[i][j]

        for i in range(0, pss.n_transf):
            for j in range(1, len(Tdis)):
                if Tdis[j-1] <= self.position_i[1][i] <= Tdis[j]:
                    x_low[1][i] = Tdis[j-1]
                    x_up[1][i] = Tdis[j]

        for i in range(0, pss.n_cap):
            hcube[0][i][0] = x_low[0][i]
            hcube[0][i][1] = x_up[0][i]

        for i in range(0, pss.n_transf):
            hcube[1][i][0] = x_low[1][i]
            hcube[1][i][1] = x_up[1][i]

        for i in range(0, pss.n_cap):
            if abs(self.position_i[0][i] - hcube[0][i][0]) <=
abs(self.position_i[0][i] - hcube[0][i][1]):
                self.position_i[0][i] = hcube[0][i][0]
            else:
                self.position_i[0][i] = hcube[0][i][1]

        for i in range(0, pss.n_transf):
            if abs(self.position_i[1][i] - hcube[1][i][0]) <=
abs(self.position_i[1][i] - hcube[1][i][1]):
                self.position_i[1][i] = hcube[1][i][0]
            else:
                self.position_i[1][i] = hcube[1][i][1]


class PSO:
    def __init__(self, power_system, x0, bounds, Qcmin, Qcmax, Tmin, Tmax, Vsmin,
Vsmax, Qcdis, Tdis, num_particles, maxiter):

        func_best_g = -1  # best fitness for group
```

```
        pos_best_g = [i[:] for i in controlvect] # best position for group
        vars = []
        p_losses = []
        new_x0 = [i[:] for i in controlvect]

        start_time = time.time();
        iterx2 = 0
        while iterx2 < maxiterx2:

            iter = 0
            swarm = []
            neigh = [[] for i in range(0, num_particles)]

            # establish the swarm:
            if iterx2 < 1:
                for i in range(0, num_particles):
                    swarm.append(Particle(x0[i]))

            # reinitialize the swarm (SO-PSO):
            else:
                for i in range(0, num_particles):
                    swarm.append(Particle(new_x0[i]))

            # begin optimization loop:
            while iter < maxiter:
                start_time_i = time.time();
                # cycle through particles in swarm
                for j in range(0, num_particles):

                    #evaluate fitness
                    swarm[j].evaluate(power_system)

                    # determine if current particle is the best (globally)
                    if (swarm[j].func_best_i < func_best_g) or func_best_g == -1:
                        pos_best_g = list(swarm[j].pos_best_i)
                        func_best_g = swarm[j].func_i

                    # find neighbourhood best particle
                    max_neigh_size = 5
                    if is_local_pso:
                        neigh[j].append(list(swarm[j].pos_best_i))
                        if iter > max_neigh_size:
                            neigh[j].remove(neigh[j][0])

                        swarm[j].get_local_best(power_system, neigh[j])

                # cycle through swarm and update velocities and position
                # and then convert continuous variables to discrete where
applicable
                for j in range(0, num_particles):
                    swarm[j].update_velocity(pos_best_g, bounds, iter, maxiter)
                    swarm[j].update_position(bounds)
                    swarm[j].discretize(Qcdis, Tdis)
                p_losses.append(func_best_g)
                vars.append(pos_best_g)

                end_time_i = time.time();
                print("Time elapsed during iteration",  iter, end_time_i -
start_time_i)

                print("Global best position", pos_best_g)
```

```
                print ("Power loss", func_best_g)

                iter += 1

            end_time = time.time();

            if sec_order == False:
                print("Total time elapsed", end_time - start_time)

            print 'FINAL:'
            print ("Qc, T, Vs", pos_best_g)
            print ("P loss", func_best_g)
            print("All p_losses", p_losses)
            print(len(p_losses))

            if sec_order == True:
                # get new initial position for the next run
                new_x0 = self.new_init_pos(pos_best_g)
                print("New initial", new_x0)

            iterx2 += 1;

        end_time2 = time.time();

        if sec_order:
            print("Total time elapsed", end_time2 - start_time)

        arr = []
        [arr.append(i) for i in range(0, len(p_losses))]
        plt.plot(arr, p_losses)
        plt.ylabel('Power loss [MW]')
        plt.xlabel('Iteration')
        #plt.show()
        #filename = "Resultados %s.png" % run
        filename = "Resultados.png"
        plt.savefig(filename)

    def new_init_pos(self, p_best_g):  # creates new initial position for next
run (second order PSO)

        minlim = 0 # minimum control variable change limit
        maxlim = 0 # maximum control variable change limit
        x0_new = [[j[:] for j in controlvect] for i in range(0, num_particles)] #
new initial position

        for n in range(0, num_particles):
            for i in range(0, num_controlvect):
                for j in range(0, len(controlvect[i])):
                    if min(p_best_g[i]) >= 0.1:
                        minlim = bounds[i][j][0] / min(p_best_g[i])
                    else:
                        minlim = bounds[i][j][0]

                    if max(p_best_g[i]) >= 0.1:
                        maxlim = bounds[i][j][1] / max(p_best_g[i])
                    else:
                        maxlim = bounds[i][j][1]

                    x0_new[n][i][j] = p_best_g[i][j] * random.uniform(maxlim,
minlim)
```

```
        return x0_new

######################## PROGRAM STARTS HERE ############################

sec_order = False
is_local_pso = False
binary_Qc = False
if pss.n_buses == 14:
    maxiter = 50
    num_particles = 60
if pss.n_buses == 39:
    maxiter = 100
    num_particles = 90

maxiterx2 = 1
if sec_order == True:
    maxiterx2 = 10

#contrl variables
global num_controlvect
num_controlvect = 3

global controlvect
global controlvect0
controlvect = [[[] for i in range(0, pss.n_cap)], [[] for i in range(0,
pss.n_transf)], [[] for i in range(0, pss.n_gen)]]
controlvect0 = [[0 for i in range(0, pss.n_cap)], [0 for i in range(0,
pss.n_transf)], [0 for i in range(0, pss.n_gen)]]
print("Control vector dimensions", controlvect)
Qc = [0 for i in range(0, pss.n_cap)]
Vs = [0 for i in range(0, pss.n_gen)]
T = [0 for i in range(0, pss.n_transf)]

x_init = [[j[:] for j in controlvect] for i in range(0, num_particles)]

if pss.n_buses == 14: # for IEEE 14 bus network
    Qcmin = 0
    Qcmax = 18
    Qcstep = 6
    Vsmin = 0.9
    Vsmax = 1.1
    Tmin = 0.9
    Tmax = 1.0999999999999994
    Tstep = 0.0125

if pss.n_buses == 39: # for IEEE 39 bus network
    Qcmin = 0
    Qcmax = 0
    Qcmax1 = 100
    Qcmax2 = 200
    Qcstep = 25
    Vsmin = 0.9
    Vsmax = 1.1
    Tmin = 0.9
    Tmax = 1.0999999999999994
    Tstep = 0.0125


#make discrete vector of capacitor sizes:
Qcdis = [[] for i in range(0, pss.n_cap)]
```

```
if pss.n_buses == 14:
    [Qcdis[0].append(i) for i in range(0, Qcmax + 1, Qcstep)]

if pss.n_buses == 39:
    [Qcdis[0].append(i) for i in range(0, Qcmax1 + 1, Qcstep)]
    [Qcdis[1].append(i) for i in range(0, Qcmax2 + 1, Qcstep)]

if pss.n_buses == 14 and binary_Qc:
    Qcdis = [[0, 18]]

if pss.n_buses == 39 and binary_Qc:
    Qcdis = [[0, 100], [0, 200]]

#get discrete vector of transformer tap positions:
Tdis = []
dis_val = Tmin

for i in range(0, pss.n_transf):
    while dis_val < Tmax+0.001:
        Tdis.append(dis_val)
        dis_val += Tstep

run = 0
num_runs = 1
p_incr = 30

#Run algorithm "num_runs" times (for testing puroposes):
for j in range(0, 1):
    print("Number of particles:", num_particles)

    for i in range(0, num_runs):

        #Initialize control variables randomly within their bounds:

        x_init = [[j[:] for j in controlvect] for i in range(0, num_particles)]

        if pss.n_buses == 14:
            for n in range(0, num_particles):
                for k in range(0, pss.n_cap):
                    if binary_Qc:
                        idx = random.randrange(0, len(Qcdis[k]), 1)
                        x_init[n][0][0] = Qcdis[k][idx]
                    else:
                        x_init[n][0][k] = random.randrange(Qcmin, Qcmax, Qcstep)

        if pss.n_buses == 39:
            for n in range(0, num_particles):
                for k in range(0, pss.n_cap):
                    if binary_Qc:
                        idx = random.randrange(0, len(Qcdis[k]), 1)
                        x_init[n][0][k] = Qcdis[k][idx]
                    else:
                        x_init[n][0][0] = random.randrange(Qcmin, Qcmax1, Qcstep)
                        x_init[n][0][1] = random.randrange(Qcmin, Qcmax2, Qcstep)

        idx = 0
        for n in range(0, num_particles):
            for k in range(0, pss.n_transf):
                idx = random.randrange(0, len(Tdis), 1)
                x_init[n][1][k] = Tdis[idx]
```

```
        for n in range(0, num_particles):
            for k in range(0, pss.n_gen):
                x_init[n][2][k] = random.uniform(Vsmax, Vsmin)

        # example: x_init = [[18], [0.978, 0.970, 0.932], [1.06, 1.045, 1.01,
1.07, 1.09]]
        print ("Capacitor sizes:", Qcdis)
        print("Transformer tap settings:", Tdis)
        print("Initial", x_init)

        #define control variable bounds:
        limits = [[Qcmin, Qcmax], [Tmin, Tmax], [Vsmin, Vsmax]]
        bounds = [[0 for i in range(0, pss.n_cap)], [0 for i in range(0,
pss.n transf)], [0 for i in range(0, pss.n gen)]]

        for i in range(0, num_controlvect):
            for j in range(0, len(controlvect[i])):
                bounds[i][j] = limits[i]

        if pss.n_buses == 39:
            bounds[0][0] = [Qcmin, Qcmax1]
            bounds[0][1] = [Qcmin, Qcmax2]

        print("Bounds", bounds)

        #create "power system" instance and run PSO:
        run += 1
        power_system = pss.PowerSystem()
        PSO(power_system, x_init, bounds, Qcmin, Qcmax, Tmin, Tmax, Vsmin, Vsmax,
Qcdis, Tdis, num_particles, maxiter)

    num_particles += p_incr

# --- END -----------------------------------------------------------------------+
```