Electrical Transmission Lines Design through Integer Multiobjective Particle Swarm Optimization Approach

Helon Vicente Hultmann Ayala*, Leandro dos Santos Coelho†, Fabio Alessandro Guerra*, Mariana Cristina Coelho*
*Power System Division (DVSE), LACTEC – Institute of Technology for Development, Curitiba, Paraná, Brazil
†Industrial and Systems Engineering Graduate Program (PGEPS). Pontifical Catholic University of Paraná, and Electrical Engineering Graduate Program (PPGEE). Federal University of Paraná, Curitiba, Paraná, Brazil
Email: hvha@ieee.org, leandro.coelho@pucpr.br, guerra@lactec.org.br, mariana@lactec.org.br

Abstract—Electrical Transmission Lines (ETL) design is normally performed aiming to reach minimum cost while satisfying project requirements. However, achieving one only design goal may not be sufficient, as the designer may need to impose other technical design metrics. This work aims to perform the ETL design task while offering the designer the possibility to impose other design goals than cost, like the ETL reliability efficiency. Multiobjective optimization algorithms are therefore suggested to perform the design task. An improved multiobjective integer version of Particle Swarm Optimization, the Integer Multiobjective Particle Swarm Optimization (IMOPSO), is proposed in this paper. The IMOPSO is applied in order to solve the multiobjective ETL design showing promising results. Conclusions are drawn regarding the IMOPSO algorithm performance and design results, through statistical analysis.

Keywords—electrical transmission lines design; multiobjective optimization; integer optimization

I. INTRODUCTION

In the past years the occurrence of blackouts in electric power transmission grids are still present, though the standards for reliability increased and the present technology of power systems has advanced. Due to the growth of interconnections and interdependence of relatively large multinational regions, Electrical Transmission Lines (ETL) network size has considerably increased [1]. This fact leads the network operators to pursue effective and economic technical solutions for failure prevention. It might be achieved by changing the network connection type and relaxed modes of operation, while maintaining competitive edge in the electricity production and transmission market. Taking into account the complexity of electrical networks, it might be difficult to perform a systematic analysis of their vulnerability and robustness to failures if one opts to infer these properties through traditional probabilistic safety assessment methods. New approaches then show themselves to be applicable in the case of electrical grid analysis, in order to infer more easily its related properties as network resistance to failure or finding the most vulnerable elements [1]–[3].

In general, the ETL design problem has a high computational cost, since the number of possible solutions grows with combinatorial computational cost when the number of transmission lines is increased. Deterministic optimization algorithms are therefore less recommended to solve the problem when applying the technique to a certain ETL size. On the other hand, metaheuristics are emerging techniques which showed in present research promising results. Evolutionary algorithms and swarm intelligence approaches are good search methods alternatives to find the optimal set of optimal solutions [4]. Particle Swarm Optimization (PSO) and its multiobjective versions have several key advantages over other existing optimization techniques in terms of simplicity, convergence speed, and robustness. Therefore the aforementioned technique has been successfully implemented in recent years to different power systems optimization problems with impressive success.

Among meta heuristic optimization algorithms, PSO has showed promising results in the past years since its original idea back to 1995 [5], also when applied to integer programming problems [6]. PSO is based on the bird flocks behavior when looking for food, where an analogy between locating food and searching global optima is proposed. Good results reported on specialized literature encouraged researchers to extend its primary version for solving multiobjective optimization problems, what resulted in many Multiobjective Particle Swarm Optimization (MOPSO) approaches [7]–[11]. In this paper, an Integer Multiobjective Particle Swarm Optimization (IMOPSO) algorithm is proposed for solving integer multiobjective optimization problems. In few words, it combines the MOPSO algorithm proposed by [8] with integer approaching techniques stated on [6], [12].

ETL design has already been solved through evolutionary multiobjective optimization algorithms [1], [2], [4]. This work focus on applying the proposed IMOPSO to solve the ETL design problem, through the use of quantitative metrics for sake of comparison analysis. Being so, we propose the ETL design procedure as a multiobjective optimization task, where the cost and reliability of the ETL network are to be
optimized. Thus, this design methodology may be applied when it is required to build a new transmission network or to perform maintenance without adding connections. The method identifies strategies for combining new lines, with different failure rates, which will be combined to increase power transmission reliability and reduce implementation costs at the same time.

The paper is organized as follows. An introduction to basic concepts of multiobjective optimization (MO) is given in Section II. In Section III, the concept of ETL MO design considering global reliability efficiency and ETL implementation cost is exposed. The IMOPSO algorithm is stated in Section IV. In Section V, the ETL design problem has its results exposed when solved by the IMOPSO algorithm, being the results then analyzed in terms of cost and reliability efficiency measures. Outcomes of the study and future research are drawn in Section VI.

II. MULTIOBJECTIVE OPTIMIZATION

A MO problem, in contrast to single-objective optimization, requires to obtain a well-distributed and diverse solution set for finding the final tradeoff in MO. One can define MO as the problem of finding a vector of decision variables that satisfies constraints and optimizes a vector function whose elements represent the objective functions. A general MO problem containing a number of objectives to be minimized and constraints to be satisfied can be written without loss of generality as [13]:

\[
\text{Minimize } f_m(x), \quad m = 1, 2, \ldots, M; \\
\text{subject to } g_j(x) \geq 0, \quad j = 1, 2, \ldots, J; \\
\quad h_k(x) = 0, \quad k = 1, 2, \ldots, K; \\
\quad x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, 2, \ldots, n.
\]

The above definition assumes \( x \) to be the vector of \( n \) decision variables: \( x = (x_1, x_2, \ldots, x_n)^T \). The last set of constraints are called variable bounds, restricting each decision variable \( x_i \) to assume a value between a lower \( x_i^{(L)} \) and an upper \( x_i^{(U)} \) bound – which define the decision variable search space \( D \). There are also \( J \) inequality and \( K \) equality constraints to be taken into account when solving a MO problem. The terms \( g_j(x) \) and \( h_k(x) \) are called constraint functions. A solution \( x \) that does not satisfy all of the \( \{ J + K \} \) constraints and all of the \( 2N \) variable bounds stated above is called an infeasible solution. On the contrary, if any solution \( x \) satisfies all constraints and variable bounds, it is known as a feasible solution. Not always the variable decision search space \( D \) is entirely feasible, what impose to MO algorithms to handle the constraints by themselves while investigating through \( D \).

In a typical multiobjective optimization problem, there exists a family of equivalent solutions that are superior to the rest of the solutions and are considered equal from the perspective of simultaneous optimization of multiple (and possibly competing) objective functions. Such solutions are called noninferior, nondominated, or Pareto-optimal solutions, and are such that no objective can be improved without degrading at least one of the others, and, given the constraints of the model, no solution exist beyond the true Pareto Front. The goal of multiobjective algorithms is to locate the whole Pareto front. Each objective component of any nondominated solution in the Pareto optimal set can only be improved by degrading at least one of its other objective components. If a solution \( x^a \) dominates \( x^b \), denoted as \( x^a \prec x^b \), the following set of \( M \) conditions, in a minimization MO problem, shall be satisfied:

\[
f_i(x^a) \leq f_i(x^b), \forall i \in \{1, 2, \ldots, M\}
\]

that is, \( x^a \) performs better objectives values than \( x^b \), when assuming the metrics stated by \( f_i \). In a MO problem, there are two main goals to be achieved: to discover solutions as close to the true Pareto-front as possible and (ii) to find solutions as diverse as possible in the obtained nondominated front.

An important issue in MO is the quantitative comparison of the performance of many runs of different algorithms. Quality measures have been introduced to compare the outcomes of multiobjective optimizers in a quantitative manner. In the case of MO evolutionary algorithms, the outcome is usually an approximation of the true Pareto Front, which is denoted as an approximation set, and therefore the question arises of how to evaluate the quality of approximation sets [14]. So, in order to evaluate different optimal Pareto sets of MO algorithms, the Spacing and Hypervolume metrics are explored in this work.

The spacing metric, hereafter denoted by \( S \), shows how well distributed the solutions are within the decision space. Conceptually, \( S \) provides information regarding the spread of the obtained non-dominated solutions. For detailed information please refer to [13]. The hypervolume metric (\( HV \)) measures the hypervolume between the estimated Pareto front \( Q \) and a reference point \( W \). \( W \) may be set as the Nadir point, which represents the opposite to that meant by an ideal point, in the context of multi-objective optimization. As the referred metric involves hypervolume calculations, there are many ways to calculate the \( HV \). In the present work \( HV \) calculations are performed through a dimension-sweep algorithm. For specific details on hypervolume computation please refer to [15]. It is obvious to notice that a higher value of \( HV \) is desired when comparing MO techniques. The greater the value of \( HV \), the greater is the hypervolume in the objective space that the output Pareto front will dominate.

A multiobjective optimization procedure outcomes a Pareto-optimal set of nondominated solutions. This fact suggests the use of an algorithm in order to calculate which is the most compromised solution among the Pareto set. As the judgment nature of the decision maker is imprecise, each objective function of the \( i \)-th solution is represented by a
The challenge in ETL design is to obtain a network that is reliable and as cheap as possible, by changing each transmission line reliability property. Such procedure is treated as a bi-objective optimization problem with conflicting objectives. For instance, if one wants to rise the network reliability, will unavoidably rise the cost, while the cost is meant to be lowered. Therefore one of the metrics studied may be calculated as [1], [16]

\[ C_{ij}(G) = \sum_{i,j \in E} \frac{w_{ij}}{\lambda_{ij}} \]

where \( w_{ij} \) is the geographical distance between vertices \( i \) and \( j \), is stated on the third column. \( w_{ij} \) are meant to be the geographical distance between each vertex. This way of treating the geographical distance in a graph has been suggested by [4]. The \( K \) matrix is defined as [4]:

\[
K^T = \begin{bmatrix}
1 & 1 & 2 & 2 & 2 & 3 & 4 & 5 & 5 & 5 & 6 & 6 & 7 & 10 & 10 & 17 & 20 & 200 & 70 \\
7 & 7 & 8 & 8 & 9 & 9 & 10 & 11 & 12 & 12 & 12 & 14 & 14 & 14 & 15 & 15 & 16 & 16 \\
2 & 4 & 3 & 9 & 4 & 5 & 6 & 9 & 10 & 10 & 10 & 60 & 60 & 60 & 50 & 50 & 50 & 50 \\
60 & 70 & 100 & 300 & 90 & 60 & 150 & 50 & 200 & 80 & 80 & 80 & 80 & 80 & 80 & 80 & 80 & 80 \\
200 & 170 & 90 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50
\end{bmatrix}
\]

Each line chosen by the design procedure has also an associated cost. The cost \( C \) is computed according to the geographical distance \( (w_{ij}) \) and the failure rate \( (\lambda_{ij}) \) between two nodes \( i, j \) as

\[ C(G) = \sum_{i,j \in E} \frac{w_{ij}}{\lambda_{ij}} \]

In order to evaluate all possible solutions to form the true Pareto-front, the number of calculations would be \( 5^{21} = 4.77 \times 10^{14} \). The number of function evaluations grows exponentially according to the number of transmission lines and types of transmission lines considered. MO techniques are then suggested to solve the problem of finding the Pareto-front in much lesser function evaluations. The goal in this ETL design example is to find solutions to populate the Pareto-front, in order to offer the designer the possibility to choose among this set a trade-off solution in a reasonable amount of computational time.

IV. INTEGER MULTIOBJECTIVE PARTICLE SWARM OPTIMIZATION

PSO is based on the concept of social learning; individuals learn by observing the successes of others. Cycling stochastically around regions of the problem space that have shown to be promising, individuals discover and share new, improved problem solutions. In a swarm, each individual embodies a candidate problem solution, and individuals interact directly with one another, gravitating toward one another’s most successful attempts at problem-solving.

As a metaheuristic algorithm emerged from collective intelligence field of study, PSO philosophy relies on the anal-
ogy between the optimization procedure and the behavior of bird flocks looking for food. From this comparison, PSO faces a bird or a fish as a particle, which represents a position within the search space, and the food that the flock looks for is the global optimum. All birds have their positions updated at each algorithm iteration, according to the influence they receive from their own past positions and the best position from the group. Being so, each particle $i$ position $x_i$ is updated according to its velocity $v_i$, specifically as

$$v_i(t+1) = w_v(t) + c_1 r_1 (pbest_i - x_i(t)) + c_2 r_2 (gbest - x_i(t))$$

(7)

$$x_i(t+1) = x_i(t) + \Delta t v_i(t+1), \Delta t = 1,$$

(8)

where $c_1, c_2$ are the acceleration coefficients, $r_1, r_2$ are random variables generated with uniform distribution in the range $[0, 1]$ and $pbest, gbest$ are the own previous best solution and the global best solution, respectively. The swarm $P$, which contains all $s$ particles, is initialized randomly. Each particle represents a candidate solution and they have information about $pbest$ and $gbest$ according to the fitness function. The $gbest$ and $pbest$ values are used to update the velocity and position equations, which will give each particle a new position at each iteration. Being so, particles fly the search space aiming to converge to the global optimum.

PSO convergence behavior is guaranteed through the use of the inertia weight $w$. The inertia weight controls not only the impact of the previous history of velocities on the current velocity update, but also the interaction between the global and local search capacity of the swarm. A large $w$ facilitates global exploration, while a small $w$ emphasizes local exploration. An appropriate value for $w$ usually assures a good trade-off between global and local search abilities, resulting in a better overall search capacity. In order to enhance the search capability of the algorithm, $w$ has a decreasing value according to the iteration number. Being so, global exploration is promoted in the beginning of the algorithm run, gradually decreasing $w$ to get more refined solutions. Thus, $w$ may be calculated as:

$$w(t) = (w_i - w_f) \frac{\text{maxiter} - t}{\text{maxiter}} + w_f$$

(9)

where $w_i$ and $w_f$ are respectively the initial and final values of $w$, maxiter is the PSO algorithm total number of iterations (generations) and $t$ is the current iteration (time).

The so-called acceleration coefficients, $c_1$ and $c_2$, are not critical parameters for PSO convergence. However, proper fine-tuning may result in faster convergence and alleviation of local minima. The parameters $r_1$ and $r_2$ are used to maintain the diversity of the population.

Integer Optimization may be formulated as $\min f(x), x \in S \subseteq \mathbb{Z}$, where $S$ is considered as the feasible region and not necessarily bounded set. Optimization techniques applied on real search spaces can be applied on such problems, in such way that they find the optimum solution by truncating the decision variable values to the nearest integer. Metaheuristic optimization algorithms on real search spaces may be implemented for solving integer programming problems. They may embed the search space $\mathbb{Z}^n$ into $\mathbb{R}^n$ and truncate the real values to integers. However, in many cases, certain problem constraints are infringed due the rounding of the solution real values and this inherently rounding procedure may result in objective function values that are far from the optimum. Nevertheless, recent research showed that the truncation of real values to integers seems not to affect significantly the performance of the PSO method [6], [12]. One possible adaptation of the PSO method to perform an integer optimization procedure, Equation (8) is changed to

$$x_i(t+1) = \text{round} (x_i(t) + \Delta t v_i(t+1))$$

(10)

where $\text{round}()$ is the truncation procedure of a real number to its nearest integer.

A. Integer Multiobjective Particle Swarm Optimization

Several approaches have been reported in the specialized literature in order to adapt the PSO algorithm to solve MO problems [7]–[11]. The main features and differences of MOPSOs approaches are in how to choose $gbest$ in order to update the swarm velocities, how to store non inferior particles in order to form the Pareto set and how to choose the project parameters from PSO.

An example of a MOPSO is the one proposed by [8], which is called Multiple Objective Particle Swarm Optimization with Crowding Distance (MOPSO-CD). MOPSO-CD utilizes the classical PSO algorithm for updating the particles velocities, in such way that the swarm flies through the search space to find the nondominated set. This set is held into an external archive that represents the approximated Pareto-set, i.e. the algorithm final result. Yet this external archive must sort the solutions contained according to its Crowding Distance (CD) factor. The CD factor is a metric that represents how dense is the region in which the particle is inserted in the objective space. The greater is the density of a solution surroundings in the objective space, the less information this solution will give to the optimization procedure. Therefore the solutions that fly through an unexplored and successful space are given more chance to influence the swarm flight.

The CD computation is proposed by [17], together with the widely used NSGA-II, a MO genetic algorithm. CD estimation for each solution requires sorting the Pareto Set according to each objective function value in ascending order of magnitude. Thereafter, for each objective function, the boundary solutions (solutions with smallest and largest function values) are assigned an infinite distance value. All other intermediate solutions are assigned a distance value equal to the absolute normalized difference in the function values of two adjacent solutions. This calculation is done for all objective functions. The overall crowding distance
begin
Initialize randomly the IMOPSO population of particles for solving $\text{min}(f_m(x))$, $m = 1, 2, \ldots, M$;
$t = 0$
repeat
Evaluate each particle: $f(x_i)$, $i \in [1, \ldots, s]$
for all particle $i \in P$ do
if $x_i \prec pbest_i$ then {Check if actual position dominates the one obtained previously by its own}
$pbest_i = x_i$
end if
if $\text{NOT}(\text{archive} \prec x_i)$ then {Check if actual position is nondominated by the external archive (archive)}
update_archive($x_i$) {Remove dominated solutions on the archive and insert $x_i$}
end if
end for
CD_sort(archive) {Sort archive on descending order according to CD values}
select_leader {Select $g_{best}$ among 10% first solutions of the CD sorted archive}
Perform PSO updates on $P$ using Eqs. (7) and (10)
$t = t + 1$
until Stopping criterion is met
end

Figure 1: IMOPSO Algorithm Pseudo-code

value is calculated as the sum of individual distance values corresponding to each objective.

Adapting MOPSO-CD to solve Integer MO problems lead to the the Integer Multiobjective Particle Swarm Optimization (IMOPSO). Figure 1 shows IMOPSO pseudo-code algorithm. Equation (8) is replaced by Equation (10), in such way that IMOPSO handles integer variables. The Pareto set is represented by the external archive, being updated at each iteration with an archiving technique. The archive is then used to define at each iteration the $g_{best}$ value, as one of the MO goals is to diversify the Pareto Front. Being so, the most diversified particles, which explore an undiscovered promising area, will have more chance to influence the velocity update procedure.

IMOPSO has one design parameter more in comparison with the classical PSO, which is the size of the external archive ($\text{archive\_size}$). This parameter sets the maximum number of particles that will form the Pareto front. When the archive becomes full and the algorithm requests to add more particles with a CD factor greater than the other particles that were already present in the archive, the CD is also used to exclude particles that have little representation for populating a dense region.

V. Simulation Results

One may formulate the ETL design as a MO procedure on the basis of $C(G)$ and $E^r(G)$ as:

$$\text{Minimize } f_m(x), m = 1, 2.$$

$$x_i \in \lambda, i = 1, 2, \ldots, |\Theta|.$$ where

$$f_1 = \frac{1}{1 + E^r(G_{x_{\text{comp}}})}, f_2 = C(G_{x_{\text{comp}}})$$

The MO problem is formulated in such way that the ETL reliability efficiency and cost are respectively maximized and minimized. The design procedure constitutes therefore an integer MO problem, where the indices of $\lambda$ are the decision variables.

Project parameters are as follows. Parameter $\text{swarm\_size} = 80$ denotes the total number of particles within the swarm, $\text{archive\_size} = 200$ is the maximal number of nondominated particles attained by archive and $\text{max\_iter} \leq 500$ is the stopping criterion. The acceleration coefficients are set as $c_1 = c_2 = 2$ and the inertial coefficient is calculated at iteration with $w_i = 0.5 , w_f = 0.3$. Simulations were done with 10 different initial random seeds. The non dominated Pareto set, obtained comparing all Pareto sets computed through all runs, contains 324 solutions and is stated on Figure 2. The most compromised solution among all non dominated solutions, according to Equation (2), is $x_{\text{comp}} = \{\lambda_i\}$ for the sequence $i = \{1, 1, 5, 5, 1, 1, 1, 3, 1, 1, 2, 4, 1, 1, 4, 1, 5, 5, 5, 5\}$. The values obtained for $x_{\text{comp}}$ returned $E^r(G_{x_{\text{comp}}}) = 0.48, C(G_{x_{\text{comp}}}) = 7.3 \times 10^3$.

It may be concluded, when comparing values presented on Table I, that the obtained spacing metric values showed the IMOPSO outcome diversity. The Hypervolume metric values, calculated for $W = \{f_1(\text{arg min}(f_2)), f_2(\text{arg min}(f_1))\}$, demonstrate a robust general outcome for many different initial conditions. The results show that IMOPSO solved the problem reasonably performing only few number of function evaluations, encouraging the application of the developed algorithm to more complex ETL design problems and another integer multiobjective problems.

VI. Conclusion

This paper presented a methodology for ETL networks design through IMOPSO. It was attempted with this configuration to obtain a set of feasible ETL networks and to offer the designer a set of possible solutions. Yet, the IMOPSO algorithm performance was assessed through the hypervolume and spacing metrics, making possible further analysis and the comparison with other MO approaches. The results showed that IMOPSO is a suitable to perform integer multiobjective optimization problems, as it performed successfully the ETL design procedure. Hypervolume and
Table I: Values Obtained After 10 Runs

<table>
<thead>
<tr>
<th></th>
<th>Spacing</th>
<th>Hypervolume</th>
<th>Pareto Front Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximum</td>
<td>46.04</td>
<td>1.884 × 10⁴</td>
<td>200</td>
</tr>
<tr>
<td>minimum</td>
<td>33.28</td>
<td>1.879 × 10⁴</td>
<td>182</td>
</tr>
<tr>
<td>mean</td>
<td>38.73</td>
<td>1.882 × 10⁴</td>
<td>196.2</td>
</tr>
<tr>
<td>standard deviation</td>
<td>4.57</td>
<td>16.63</td>
<td>5.57</td>
</tr>
</tbody>
</table>

Figure 2: Pareto front (dot points). \((f_1(x_{\text{comp}}), f_2(x_{\text{comp}}))\) is represented by a square. Solutions with minimum \(f_i; i = 1, 2\) by a circle and the Nadir point \(W\) is denoted by a star.

spacing metrics, obtained in 10 runs with different initial conditions, showed the IMOPSO algorithm robustness and capacity to obtain the Pareto front and diversify it – the main goals in multiobjective optimization. Future research focus on the implementation of more complex networks and test different metaheuristics algorithms on the exposed benchmark.

REFERENCES


