Abstract—This paper proposes a method to solve the long-term scheduling of the Brazilian hydroelectric power system. The objective is to meet the energy demand by defining the required actions for each individual power plant, whilst respecting boundary and operational constraints. The mathematical model is formulated as a dynamic, nonlinear, multiobjective, high dimensional and largely constrained optimization problem. We compare two classes of metaheuristics for constrained problems: Differential Evolution (DE) and swarm-based (PSO), using the CEC-2017 constrained optimization competition benchmark. The best performing metaheuristics of each class, evolutionary-based LSHADE44b (a slight variant of LSHADE44, winner of CEC-2017) and swarm-based EPSO-G, are then compared on the hydroelectric power system scheduling problem. We considered 111 power plants for a period of 5 years, with monthly time-step, resulting in 13,320 decision variables with 20,091 constraints, which are optimized for 194 random affluence scenarios. Results show that LSHADE44b outperforms EPSO-G in all scenarios, reducing the constraints violations down to an average of 16. On 45% of the affluence scenarios there were no violations, and the cost steadily decreased, showing convergence after the constraints are satisfied.

Index Terms—hydroelectric dispatch planning, high dimensionality, highly constrained problem, differential evolution, particle swarm optimization

I. INTRODUCTION

The planning/scheduling of the hydroelectric power generation system is a high dimension, highly constrained, real-world optimization problem of great economical and environmental importance in many countries, specially in Brazil, which has hydroelectric power plants as its main energy source. In the long-term operation scheduling, a time frame of 60 months (5 years) is considered, and the plants are usually grouped into five subsystems, each being treated as one single power plant [1]. However, this is an oversimplification of the problem which hides many of its important non-linear features.

In contrast to the unified sub-system approach, the model used for this paper [2], [3] comprehends monthly operational actions for each hydroelectric power plant, defining the amount of discharged water (going through generation units) and spilled water for each individual plant, considering its hydrographic basin interconnections to other plants. The aim is to minimize the energy deficit whilst attending to boundary constraints (maximum flow and spill), and to operational constraints (reservoir volume limits and minimum flow). The energy deficit is computed for each sub-system (as demands are only available for sub-systems), and the deficit cost is modeled as a nonlinear function of the energy deficit.

Two equations are important for the optimization: the deficit function and the hydraulic balance. The deficit function relates energy demand with the energy generation capacity, considering net water height, water discharged, productivity and availability of each plant. The hydraulic balance determines the reservoir’s final volume from the initial volume, water inflow and water outflow.

Another important aspect that strongly influences the solution of the planning problem are the affluence scenarios. Greater affluence leads to more viable solutions and less energy deficit. In order to achieve good confidence on planning results for future operations, multiple synthetic affluence time series, generated by stochastic models capable of reproducing historical statistics and preserving the spatial correlation between the power plants are required [4].

The objective of this work is to obtain viable solutions for the hydroelectric dispatch planning problem, using swarm and evolutionary based metaheuristics adapted to constrained problems. The benchmark CEC-2017 [5] is used to assess the capacity of the tested metaheuristics in handling constrained problems, and the best performer of each class is applied to the optimization of the dispatch planning problem. The main contribution of this paper is to show that the proposed methods are suitable for finding viable solutions in a high dimensional real world problem with 111 hydroelectric power plants from the Brazilian energy matrix, resulting in a decision vector of 13,320 dimensions and 20,091 constraints.

II. LITERATURE REVIEW

Given the importance for real-world problems, algorithms for constrained optimization comprise great part of the metaheuristic literature. Specifically in this work, two important categories of algorithms are used: 1) variants of the classic swarm algorithm PSO, due to its good capability of obtaining good results in short processing time; and 2) variants of the evolutionary algorithm DE, given its versatility with promising results in recent literature for a broad category of problems.

In [6], the techniques death penalty, which removes non-viable solutions, and penalty-based fitness, which composes the fitness with static values according to the violated constraint are compared. Experiments show that the penalty-based fitness was superior in nearly half of the functions tested, matching death-penalty’s results on remaining functions.

The work developed in [7] uses the algorithm MPSO (Mutation Particle Swarm Optimization) to optimize the problem of...
the optimal operation of thermoelectric power plants. MPSO applies a mutation operator on the particles that provide better diversity, and classify the particles into two categories, one for global and one for local optimization. This classification process contributes to maintain the diversity while promoting faster convergence. Dynamic penalties were applied by multiplying the fitness by the number of violated constraints. The MPSO was compared to 13 optimization algorithms (e.g. PSOLR [8] (PSO Lagrangian Relaxation) and MILP [9] (Mixed-Integer Linear Programming), obtaining the best results.

The $\epsilon$-constrained technique [10] prioritizes constraint satisfaction over the fitness value $f$, using the $\beta$-value metric that is usually calculated as the weighted sum of the violated constraint values. A relaxation mechanism allows the comparison of two solutions $p_1$ and $p_2$ if their $\beta$-value is smaller than an $\epsilon$ value (Equation 1). A linear decrease can be applied to $\epsilon$, increasing the importance of constraint satisfaction, and $\epsilon = 0$ gives the highest importance to constraint resolution.

$$p_1(f_1, \beta_1)<p_2(f_2, \beta_2) \Leftrightarrow \begin{cases} f_1 < f_2, & \text{if } \beta_1, \beta_2 \leq \epsilon \\ f_1 < f_2, & \text{if } \beta_1 = \beta_2 \\ \beta_1 < \beta_2, & \text{otherwise} \end{cases} \quad (1)$$

The $\epsilon$-constrained technique is applied on the PSO ($\epsilon$PSO) in [11], aiming to reduce the number of violated constraints and improve the fitness of solutions that do not violate constraints. The initial value of the relaxation $\epsilon$ is the average of the violated constraints at the first generation, which is gradually decremented to zero when 80% of the iterations are reached. The $\epsilon$PSO was compared with the GENOCOP 5.0 algorithm [12] (a GA variant for constrained problems), with $\epsilon$PSO presenting superior and faster results in all experiments.

A mechanism to control the velocity of the particles was added to $\epsilon$PSO in [13] in order to prevent particles of leaving viable regions. The particles are split into $K$ groups, which are classified according to the number of viable particles in each group. The velocity of the group with less viable particles is then limited by the velocity of the group with more viable particles. Experiments comparing this variant with the original $\epsilon$PSO showed that the proposed mechanism to control velocity obtained the best results in shorter processing time.

In [14], the $\epsilon$-constrained technique was applied on the DE algorithm using no relaxation ($\epsilon = 0$). Two additional mechanisms were added to $\epsilon$DE in [15]: the gradient mutation, a time-consuming repair strategy; and the elite preservation; which keep the solutions with less violations for mutation and crossover. Experiments comparing $\epsilon$DE with $\epsilon$PSO and $\epsilon$GA showed that $\epsilon$DE obtained better results, with less fitness function evaluations for all tested problems.

A hybrid EPSO-G (PSO algorithm with gradient mutation, a new velocity updating rule, and $\epsilon$-level constraint handling) with local search CMA-ES [16] (Covariance Matrix Adaptation Evolutionary Strategy), namely EPSO-CMA, is presented on [17]. EPSO-CMA evolves multiple independent swarms, mutating the velocity to prevent stagnation, treating constraints with the $\epsilon$-constrained technique, and using the best particle from all swarms as the global best result. The proposed hybrid method was then compared with $\epsilon$DEag and Co-CLPSO [18] (Coevolutionary comprehensive learning PSO, that uses two swarms and assigns different particles to evaluate distinct constraints). Results showed that EPSO-CMA obtained 100% of viable solutions in all tested functions, whereas Co-CLPSO obtained viable solutions in only part of the functions, and $\epsilon$DEag did not find viable solutions.

Given its high versatility, the most successful algorithms for constrained optimization problems are DE based. The CEC 2017 [5] is a competition on constrained optimization that provides 28 constrained optimization problems, whose best ranked algorithms are LSHADE44 [19] (Linear Success History Adaptive Differential Evolution), UDE [20] (Unified Differential Evolution), LSHADE44 + IDE [21] (LSHADE44 + Individual-DEpendent Mechanism) and CAL-SHADE [22] (Constraint Handling With Success History Adaptive Differential Evolution), all DE based algorithms.

III. LONG TERM PLANNING OF HYDROELECTRIC DISPATCH

The Brazilian hydroelectric dispatch planning for long term operations aims to create a plan for all U power plants of the system for a monthly period of $T$ months. These power plants are divided into $S = 5$ interconnected subsystems, each one with its own energy demand ($D_{s,t}$) and deficit cost coefficients ($\lambda_1, \lambda_2, \lambda_3$). The plan (solution vector) $x = [QC_{u,t}, QV_{u,t}, (u = [1, U], t = [1, T])$ contains, for each power plant, the monthly amount of discharged water ($QC_{u,t}$) and spilled water ($QV_{u,t}$), and should minimize the average operating cost (Custo Médio Operacional - CMO) objective function $f(x)$ (2). Two subproducts of a solution plan are the amount of energy deficit ($D_{s,t}$) (3) and the volume of the reservoirs ($V_{u,t}$) (6).

$$f = \sum_{t=1}^{T} \lambda_t \left( \sum_{s=1}^{S} \left( \lambda_{1s} D_{s,t} + \lambda_{2s} (D_{s,t})^2 \right) \right) \quad (2)$$

$$D_{s,t} = \left( DEM_{s,t} - \sum_{u=1}^{U_s} (H_{u,t} \times QC_{u,t} \times Prod_u \times Ind_u) \right) \quad (3)$$

subject to boundary constraints [23]:

$$0 <= QC_{u,t} <= QC_{u}^{max} \quad (4)$$

$$0 <= QV_{u,t} <= QV_{u}^{max} \quad (5)$$

where:

- $\lambda_t$ is a time dependant and $\lambda_{1s}, \lambda_{2s}$ are sub-system dependant coefficients for the cost of energy deficit;
- $DEM_{s,t}$ is the energy demand;
- $U_s$ is the number of plants in subsystem $S$;
- $H_{u,t}$ is the net height of the water column;
- $Prod_u$ is the productivity of the power plant;
- $Ind_u$ is the programmed unavailability;
The resulting volume of the water reservoirs is given by:
\[
V_{u,t} = V_{u,t-1} - QV_{u,t} - QC_{u,t} + A_{u,t} + \sum_{m \in M_u} (QV_{m,t} + QC_{m,t} - A_{m,t})
\]
where \(M_u\) is the set of power plants immediately upstream of \(u\) and \(A_{u,t}\) is the natural incremental water influence to the reservoir.

The operational constraints regard the percentage \(\rho\) of the maximum reservoir volume at the last month:
\[
\rho V_{u,max} \leq V_{u,T}
\]
the reservoir volume limits \([V_{u,min}, V_{u,max}]:
\[
V_{u,min} \leq V_{u,T} \leq V_{u,max}
\]
and the minimum required water flow \((Q F_u):
\[
QF_u \leq QC_{u,t} + QV_{u,t}
\]
A solution plan \(x\) violates the constraints if \(\beta(x) > 0:\)
\[
\beta(x) = \sum_{i=1}^{T} \sum_{u=1}^{U} \left( \max(0, QF_u - QC_{u,t} - QV_{u,t}) + \max(0, V_{u,min} - V_{u,t}) + \max(0, V_{u,t} - V_{u,max}) \right)
\]
\[
+ \sum_{u=1}^{U} \left( \max(0, \rho V_{u,max} - V_{u,T}) \right)
\]

IV. LSHADE with \(\epsilon\)-Constrained Meta-heuristic

The algorithm LSHADE44 is a DE variant that includes mechanisms to auto-adjust the parameters, to select the best mutation strategy and to maintain the population diversity. It was initially proposed by [24] and later adapted to optimize constrained problems [19], [25], using the \(\epsilon\)-constrained mechanism [10] to treat the constraints, prioritizing constraint resolution over cost function minimization.

The \(\epsilon\)-constrained is used to compare an individual \(x_i\) with its offspring \(u_i\) (Algorithm 1, line 10), to compare individuals when linearly reducing the archive \(A\) and population size to a minimum size (line 21), and to define the group of the best individuals. The archive \(A\) stores, at every iteration, the worst individual between \(x_i\) and \(u_i\), up to a maximum limit, and once the limit is reached, a random individual from the archive \(A\) is replaced by the newly added individual. The function \(\text{evaluate}()\) (lines 2 and 10) assesses not only the objective function, but also the problem constraints.

The category of SHADE algorithms [26] auto-ajust the main parameters by using a circular memory that stores a success history. The LSHADE44 generates values for the parameters \(F\) (scale factor of the differential variation) and \(CR\) (crossover rate - Algorithm 1, line 7) using a memory comprised of 4 pairs of vectors. At every iteration, a random vector is selected, and the value from a sequential position (in round-robin) is taken from the vector. This value is used to select, from a Cauchy distribution with standard deviation 0.1, the next value for \(F\). The parameter \(CR\) is defined in the same way, but using a normal distribution. The best values for \(F\) and \(CR\) are then used to update the vectors \(B_F\) and \(B_CR\) (lines 14 and 15, respectively), which in turn are used to update the memory vector (line 22).

The four mutation strategies used by LSHADE44
\[
q = [(\text{current} - \text{to-pbest/1/bin}), (\text{randr/1/bin}), (\text{current} - \text{to-pbest/1/exp}), (\text{randr/1/exp})]
\]
aim to promote diversity. A strategy is selected from \(q\) at every iteration based on its probability \(q_l, l = [1, 4]\), which is initialized with \(q_1 = \frac{1}{4}, q_2 = \frac{1}{4}, q_3 = \frac{1}{4}, q_4 = \frac{1}{4}\) and increased when the strategy is successful (Algorithm 1, line 13). To avoid that a specific mutation strategy dominates, \(q_l\) is reset to its starting values whenever a probability decreases below the given limit \(\delta > 0\).

The mutation strategies \(\text{current} - \text{to-pbest/1/bin}\) and \(\text{current} - \text{to-pbest/1/exp}\) compute the trial vector \(x_{new,i}\) according to the equation (11):
\[
 x_{new,i} = x_i + F(x_{pbest} - x_i) + F(r_1 - r_2), \quad (11)
\]
where \(x_i\) is the current individual, and \(x_{pbest}\), \(r_1\) and \(r_2\) are three randomly chosen individuals selected from the \(p_{best}\) best individuals, from the population, and from the population joined with the archive \(A\) (respectively).

The mutation strategy \(\text{randr/1} (12)\) is a variant of the traditional \(\text{randr/1}\) mutation [27], which randomly selects

---

**Algorithm 1 LSHADE44**

1. Initialize: \(x\), memories, \(q_l = [\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}]\), \(l = [1, 4]\), \(\text{archive} A = \emptyset\), \(N = N_{initial}\)
2. evaluate(\(x\))
3. repeat
4. \(B_F = \emptyset\), \(B_CR = \emptyset\)
5. for all \(x_i\), in \(x\) do
6. \(\text{Select a mutation strategy } l \in q\) with probability \(q_l\)
7. \(\text{Generate } F\) and \(CR\) using memories
8. \(x_{new,i} \leftarrow \text{mutated } x_i\)
9. \(u_i \leftarrow \text{crossover between } x_{new,i} \text{ and } x_i\)
10. if \(\text{evaluate}(u_i) < \epsilon\)-constrained \(\text{evaluate}(x_i)\) then
11. \(x_{new} \leftarrow x_{new} + u_i\)
12. \(\text{archive} A \leftarrow \text{archive} A + x_i\)
13. \(\text{Update probabilities and mutation strategy } q_l\)
14. \(B_F \leftarrow B_F + F\)
15. \(B_CR \leftarrow B_CR + CR\)
16. else
17. \(x_{new} \leftarrow x_{new} + x_i\)
18. \(\text{archive} A \leftarrow \text{archive} A + u_i\)
19. end if
20. end for
21. \(x \leftarrow \text{best } N\) individuals from \(x_{new}\)
22. Update memories using \(B_F\) and \(B_CR\)
23. Update \(N\) \(\rightarrow \text{adjust population size}\)
24. until exit condition
three individuals $r_1$, $r_2$ and $r_3$ from the population, and sets the best selected individual as $r_1$.

$$x_{\text{new},i} = r_1 + F(r_2 - r_3)$$

(12)

The source code of the proposed LSHADE44b implementation is available for download at [28].

V. EXPERIMENTAL RESULTS

A. Benchmark of Proposed Solution

The variant of the LSHADE44 used in this work, namely LSHADE44b, is a reimplementation of the algorithm presented in [19], with the differences motivated by the fact that the reference paper does not provide an implementation nor completely define all implementation details. There is no significant modification, but whenever our implementation is used, it is referenced as LSHADE44b. When results from the original version are reported, we reference it as LSHADE44.

Given that LSHADE44 does not specify a handling mechanism for boundary constraints, LSHADE44b uses the conventional repair strategy, which replaces the position that violates the boundaries by the maximum (or minimum) value of the violated dimension. LSHADE44b maintains the size of the archive A proportional to the population size (which has a linear decrease) by removing the worst individual from the file, using the $c$-constrained for the comparisons.

The competition benchmark CEC 2017 [5] (Special Session and Competition on Constrained Real-Parameter Optimization) is used to test the performance of the proposed variant LSHADE44b. This benchmark is composed of 28 constrained functions of $d = \{10, 30, 50, 100\}$ dimensions [29]. To compare LSHADE44b with state-of-the-art algorithms for constrained problems, results for the following algorithms are also provided: LSHADE44, EPSO-G, EPSO (PSO with $c$-constrained), SPSO (PSO with static penalty) and DE with $c$-constrained. The algorithm LSHADE44b is developed in C++11, using the open source scientific library PaGMO/PyGMO for evolutionary algorithms [30], whereas EPSO-G is developed in Python 3.5.

For LSHADE44b, the parameter values defined in [21] are used: initial population size $p = 18 \times d$, linearly decreasing up to $p = 4$; size of archive $A$ as $2.6 \times p$; best individuals’ group size $p_{\text{best}} = 0.11 \times p$; vectors $F$ and $CR$ with 6 positions each, initialized with the value 0.5. For EPSO-G, the parameter values are defined as: $p = 30$; inertial weight $w = 1$, linearly decreasing up to 0.2; random initial velocity $v \leq 5\%$ of the search space; social and cognitive weights $c_1 = c_2 = 1$; $\gamma = 1 \times 10^{-1}$; and $c = 1/d^2$. Both algorithms use $c$-constrained with $\epsilon = 0$.

Table I presents the results of the developed methods and includes the results for the algorithms LSHADE44, UDE, LSHADE+IDE and CAL-SHADE as reported on [5] (they were not implemented in this work). The bold cells show the best result for each dimension, and the algorithms’ overall results are presented on the column Score. The scores are based on the mean and median results, considering (in this order) the number of viable solutions, the violation values and the objective function.

As can be seen on Table I, the proposed modifications on the algorithm LSHADE44b, like the automatic reduction of the archive A and the treatment of the boundary problems resulted in better performance than the original LSHADE44 on lower dimensions. On the other hand, LSHADE44b showed slightly worse results than LSHADE44 on higher dimensions, whereas LSHADE44 presented a more stable behavior. However, given that the differences are not substantially relevant, and that the original implementation of LSHADE44 was not publicly available at the time, LSHADE44b was used for the remaining experiments of this work. The second best algorithm developed in this work, EPSO-G, is used as a measure of comparison to assess LSHADE44b’s overall performance.

B. Hydroelectric Dispatch Planning

Two scenarios of initial reservoir volume were used to test the algorithms EPSO-G and LSHADE44b on the dispatch planning problem: a volume conservation scenario (VCS) with initial volumes at 70% of the maximum capacity; and a volume recovery scenario (VRS) with initial volumes at 30% of the maximum capacity. The final volume of the reservoirs was predefined at 70% of the maximum capacity for both scenarios ($\rho = 0.7$) in order to avoid “end of the world” plans, where all water is spent in the last months of the simulation.

Each algorithm was executed for every scenario over 194 different synthetic affluence series for a system of 111 hydroelectric power plants. This resulted in an optimization problem with $d = 13,320$ dimensions (111 power plants $\times$ 60 months $\times$ 2 decision variables) and $d = 20,091$ constraints (111 power plants $\times$ 60 months $\times$ (max/min reservoir volumes + minimum water flow) + 111 final reservoir levels). Stopping criteria for the metaheuristics are met when the difference between the costs or between the solution vectors of the population’s best and worst individuals is less than $10^{-6}$ (limited to 500K objective function evaluations).

A repair strategy is applied to the solution vector whenever reservoir volume constraints are not fulfilled (8). The strategy consist in first decreasing the spilled water ($Q_{V_{u,t}}$) and then the amount of discharged water ($QC_{u,t}$), until the volume reaches a valid level, or these variables are at their minimum.
Configuration for LSHADE44b parameters are: population size \( p = 100 \) linearly decreasing to \( p = 4 \); size of archive \( A \) is \( 2.6 \times p \); best individuals’ group size \( p_{best} = 0.11 \times p \); vectors \( F \) and \( CR \) with 6 positions each, initialized with the value \( 0.5 \). EPSO-G configuration parameters are: \( p = 120 \); \( c_1 = c_2 = 0.8 \); random initial velocity \( v \leq 5\% \) of the search space; \( \gamma = 1 \times 10^{-10} \); and \( c = 1/d^2 \). Both algorithms use \( \epsilon \)-constrained with \( \epsilon = 0 \).

### Table II

<table>
<thead>
<tr>
<th>Level</th>
<th>( \beta(x) ) and corresponding total number of violated constraints; non-zero components of ( \beta(x) ) (volume and minimum flow), and cost function (fitness), considering initial reservoir volumes at 70% (VCS) and 30% (VRS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta(x) )</td>
<td># Viol. Constr.</td>
</tr>
<tr>
<td>VCS 70%</td>
<td>mean</td>
</tr>
<tr>
<td></td>
<td>worst</td>
</tr>
<tr>
<td></td>
<td>best</td>
</tr>
<tr>
<td>VRS 30%</td>
<td>mean</td>
</tr>
<tr>
<td></td>
<td>worst</td>
</tr>
</tbody>
</table>

Table II shows the \( \beta \)-Violation values (10); the total number of violated constraints; components of \( \beta \)-Violation (violations on final volume (7) and on the minimum water flow (9)); and the solutions costs (2). Due to the repair strategy, no solution violated the maximum and minimum volume constraints, hence the respective values are omitted from the table.

As we can see on Table II, LSHADE44b presents superior mean results in all cases, obtaining smaller costs and violating nearly 10 times less constraints than EPSO-G. When comparing the mean \( \beta \)-Violation values, LSHADE44b performs up to 3 orders of magnitude better than EPSO-G. As expected, for both algorithms the mean number of violated constraints is lower for the VCS scenario than for the VRS scenario, probably due to the greater water availability on the start.

Although the worst results have high \( \beta \)-Violation values, the number of violated constraints is not high (0.986%), showing that the algorithms are able to solve most of the problems constraints. From the problem’s 20,091 constraints, none were violated on 88 of the 194 affluence series for the VRS scenario (45%), and on 83 of the series for the VRS scenario (42%). The best results of LSHADE44b, for both scenarios show that once all constraints are solved, the algorithm successfully optimizes the cost of the solution.

In all cases, when analyzing the ratios between the mean results and the standard deviations (\( \sigma \)), LSHADE44b presents high ratios whereas EPSO-G shows low ratios. However, given the higher \( \beta \)-Violation values of EPSO-G compared to LSHADE44b, it can be concluded that the small ratios obtained by EPSO-G are due to the fact that it has a poor convergence on the hydroelectric dispatch problem, so that the mean EPSO-G result is equally bad at the task, whereas LSHADE44b presents a good convergence, and its high standard deviations is due to the differences in the affluence series.

Figure 1 shows the average convergence of LSHADE44b throughout generations, presenting both the cost optimization (solid lines) and the \( \beta \)-Violations (dotted lines) for two cases: the average results for the 30 best solutions (AvgBest, in blue) and the average results for the 30 worst solutions (AvgWorst, in red). The VCS and VRS scenarios are presented in Figure 1(VCS) and Figure 1(VRS).

In both AvgBest and AvgWorst cases the figures indicate a good convergence of the \( \beta \)-Violations, with AvgBest converging to zero (i.e., no violated constraints) and AvgWorst converging to a small number of violations. But even this small \( \beta \)-Violation prevents cost improvements, since violations are prioritized over cost when computing the fitness. Nonetheless, good cost improvements can be seen on AvgBest cases, given that a totally viable population is obtained around 8000 generations, demonstrating the positive effect of the \( \beta \)-Violation prioritization over the cost.

The best final cost was quite unexpectedly obtained by the VRS scenario (Figure 1(VRS)) and might be explained by its slower \( \beta \)-Violation convergence. The fact that this scenario has less water available at the initial periods causes a larger number of nonviable solutions, forcing an increase in the populations diversity. This allows a better exploration of the search space with possibly better viable results in the long term. The more water abundant scenario (VCS) (Figure 1(VCS)) can be leading to a quicker population displacement towards a stable region of the search space (local minima).

### VI. Conclusion

Despite being much referenced in the literature, swarm based metaheuristics proved to have far worse results than differential evolution based metaheuristics on high-dimensional constrained problems. The proposed implementation of the CEC-2017 competition winner, LSHADE44b, surpassed all swarm-based and even the original version on most of the benchmark problems (Table I).

LSHADE44b successfully optimized the hydroelectric dispatch planning problem with 13,320 dimensions, without violating any of the 20,091 constraints on 88 of the 194 affluence series for the VCS scenario (45%), and on 83 of the series for the VRS scenario (42%). On these cases, cost improvements occurred throughout the generations and convergence continued for a long time without new constraint violations. In the worst ranked solutions, LSHADE44b was still able to reduce the constraints violations to nearly zero, although its constraint handling mechanism did not allow it to significantly improve the costs.

### Acknowledgement

This project was partially supported by the ANEEL P&D Project PD-06491-0307-2013 proposed by Copel GeT.
Fig. 1. Average Cost (2) (solid lines) and β-Violations (10) (dotted lines) per generation for the hydroelectric dispatch planning problem solved with the LSHADE44b metaheuristic. Reservoir initial volume scenarios at 70% (VCS) and at 30% (VRS) of the total capacity. AvgBest is the average of 30 best solutions (blue lines) and AvgWorst is the average of the 30 worst solutions (red lines).

REFERENCES


