Intelligent Algorithms applied to Wind Turbine Control

Fernando Oterino

Electronic Technology Department

University College of Engineering at Vitoria-Gasteiz,

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Escuela Universitaria de Ingeniería Vitoria-Gasteiz

Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

INTRODUCTION

- The Evolutionary Algorithms (EA) and the Swarm Intelligence (SI) are different fields inside of the Artificial Intelligence developed from the 70’s.

- The principal target is generate solutions to difficult optimization problems using techniques inspired in the natural evolution, such as mutation, selection, and crossover.

Evolutionary Algorithms

- In a Evolutionary Algorithm, we start with a randomly population of the candidate solutions (Agents) to an optimization problem.
- The agents may be of one or more dimensions (chromosomes) depends on the dimension of the search space
- The initial population is improved it through repetitive application of the mutation, crossover, and selection operators.
- In each generation, the agents with worst result in the objective function are discarded

Evolutionary Algorithms

- The usual steps in a EA are:
  - ***Initialization:*** the initial population is generated, the principal factors in this step is the size of the initial population, number of chromosomes by agent, and decide if the initial population is random or not.
  - ***Genetic operators:*** the creation of a new generation is done by mutation and crossover of different agents and even different chromosomes, each new agent can be generated by two or more agents of the previous generation.
  - ***Selection:*** compare the new and the old generation and will survive the agents with best response in a objective function desired, the new generation is a mixture of previous generations with the best agents of each generation.
  - ***Termination:*** This generational process is repeated until a termination condition has been reached

Swarm Intelligence

- In the swarm intelligence, a population of candidate solutions (agents) moves in the search space, this movement can be a random movement or guided movement by different parameters.
- In each new position, the objective function is evaluated and we decided if the new position is better or not

Swarm Intelligence

- The usual steps in a SI are:
  - ***Initialization:*** the initial population is generated, the principal factors in this step is the size of the initial population, number of chromosomes by agent, and decide if the initial population is random or not.
  - ***Movement:*** each agent is moved to a new position where the objective function is evaluated and it is compared with the previous position, the final position will be the position with better result in the objective function
  - ***Termination:*** This generational process is repeated until a termination condition has been reached

Examples of Evolutionary Algorithms

- ***<u>Differential Evolution (DE)</u>*** *Storn, R.; Price, K. (1997). "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces". Journal of Global Optimization **11**: 341–359.*

- ***<u>Backtracking Search Algorithm (BSA)</u>*** Civicioglu, Pinar(2013) "*Backtracking Search Optimization for numerical optimization problems ",* Applied Mathematics and Computation Volume: 219 Issue: 15 Pages: 8121-8144.

Examples of Swarm Intelligence

- ***Particle Swarm Optimization (PSO)*** Cagnina, L., Esquivel, S., Gallard R. “CEC2004: Proceedings of the 2004 Congress on Evolutionary Computation, Vols 1 and 2

Differential Evolution

The basic DE algorithm can be described with the following steps:

- Initialization.
- Mutation
- Recombination
- Selection

Differential Evolution

Initialization:

1. Initialize the initial population of agents (x) with random values in the search-space.

First Generation

Minimum acceptable value of m-chromosome

$$X_{p,m}^{1} = X_{m}^{\min} + rand(0,1) \cdot (X_{m}^{\max} - X_{m}^{\min})$$

"m" chromosome of the "p" agent

Agent "p" of a population of Np agents

- Np = Size of the population
- m = 1...n with n=Dimension of thet searc space

Differential Evolution

Mutation:

1. Construction of Np new random agents using three agents "Xa", "Xb" and "Xc" presents in the actual generation, these three agents must be distinct from each other.

$$Y_p^{\overset{\text{Generation}}{g}} = X_c + F \cdot (X_a - X_b)$$

$$F \in [0,2];\ \text{Mutation Rate}$$

Differential Evolution

Recombination:

1. With the actual generation agents and the agents obtained in the mutation step, the next step is the recombination with the following expression

$$T^g_{p,m} = \begin{cases} Y^g_{p,m} & \text{if rand(0,1)<G.R.} \\ X^g_{p,m} & \text{otherwise} \end{cases}$$

$$G.R. \in [0,1] \text{ ;Recombination Rate}$$

- The comparison is made chromosome by chromosome, so the test agent will be a mix of "x" and "y" agents.

Selection:

Finally, the selection is made comparing the "T" agents and the original agents, so that the survival agent will be the agent with better result in the objective function.

This generational process (Mutation, Recombination and Selection) is repeated until a termination condition has been reached

Backtracking Search Algorithm (BSA)

The basic BSA algorithm can be described with the following steps:

Initialization.

Mutation

Crossover

Selection I

Backtracking Search Algorithm (BSA)

Initialization:

1. Initialize the initial population of agents (x) with random values in the search-space. the value of each chromosome in a agent is obtained with a uniform distribution

$$X_{i,j} = U(low_j, up_j)$$

i :1.. Np ; Np =Size of the population.

j:1..m; m = Dimension of search space

U: Uniform distribution

2. In the same way a second population is initialized to be used as historical population

$$oldX_{i,j} = U(low_j, up_j)$$

* In every interaction, we will randomly work with X or oldx

Backtracking Search Algorithm (BSA)

Mutation:

BSA´s mutation step generates one Mutant population (Matrix Y) with the following expression:

$$Y = X + F \cdot (oldX - X)$$

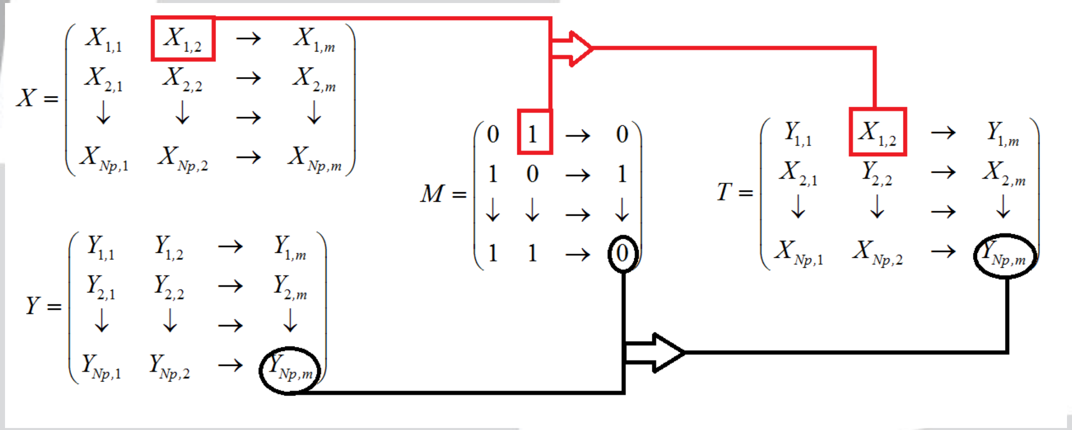
$$F = 3 \cdot N(0,1); \text{ Mutation Rate}$$

- Unlike the differential evolution in this mutation is considered agents of previous generations

Backtracking Search Algorithm (BSA)

Crossover:

- BSA’s crossover process generates the final test population matrix T. In this step, a random binary matrix (MAP) is generated, the dimension of this matrix is Np X m (size of the population X dimension of the search space).
- the elements of the matrix T are formed by the elements of the matrix X in the coordinates in which MAP = 1, in the other positions have the value of the matrix Y



$X =$
$X_{1,1}$ $X_{1,2}$ $X_{1,m}$
$X_{2,1}$ $X_{2,2}$ $X_{2,m}$
$X_{Np,1}$ $X_{Np,2}$ $X_{Np,m}$
$Y =$
$Y_{1,1}$ $Y_{1,2}$ $Y_{1,m}$
$Y_{2,1}$ $Y_{2,2}$ $Y_{2,m}$
$Y_{Np,1}$ $Y_{Np,2}$ $Y_{Np,m}$
$M =$
0 1 0
1 0 1
1 1 0
$T =$
$Y_{1,1}$ $X_{1,2}$ $Y_{1,m}$
$X_{2,1}$ $Y_{2,2}$ $X_{2,m}$
$X_{Np,1}$ $X_{Np,2}$ $Y_{Np,m}$

- X = Current Generation Matrix, Xij=Chromosome “j” of “i” agent.
- Y = Mutant Matrix, Yij=Chromosome “j” of “i” agent.
- M= MAP Matrix example.
- T= Test population matrix

Backtracking Search Algorithm (BSA)

Selection:

- Finally, the selection is made comparing the "T" agents and the original agents, so that the survival agent will be the agent with better result in the objective function.

This generational process (Mutation, Crossover and Selection) is repeated until a termination condition has been reached

Particle Swarm Optimization(PSO)

The basic PSO algorithm can be described with the following steps:

Initialization.

Initial assessment

Update speeds and positions

Assessment

Particle Swarm Optimization(PSO)

Initialization:

Initialize the initial population of agents (x) with random values in the search-space. The value of each chromosome in a agent is obtained with a uniform distribution

$$X_{i,j} = U(low_j, up_j)$$

i :1..Np ; Np =Size of the population.

j:1..m; m = Dimension of search space

U: Uniform distribution

Initialize the initial speed of each agent (V) with random values in the search-space. Each agent will have a speed for each chromosome

$$V_{i,j} = U(low_j, up_j)$$

Particle Swarm Optimization(PSO)

Initial assessment :

Each agent is evaluated with the objective function and the is searched the agent that has obtained the best result

Update speeds and positions :

Update Speeds: the speed of each agent is updated with the following step:

Random (0.5,1)

$$V_{i,j}(t) = W \cdot V_{i,j}(t-1) + \ell_1 \cdot (P_{ij} - X_{ij}(t-1)) + \ell_2 \cdot (P_{b,j} - X_{ij}(t-1))$$

Inertia : random(0,1)

The best historical position of the Xij agent

Velocity, agent “i”, Chromosome “j”

The best historical position of all agents

Update Positions: the speed of each agent is updated with the following step:

$$X_{i,j}(t) = X_{i,j}(t-1) + V_{i,j}(t)$$

Particle Swarm Optimization(PSO)

Assessment :

Evaluate the objective function for the new positions reached by the agents and the values of "Pij" and "Pbj" will be actiualized

This generational process (Update speeds and positions, Assessment) is repeated until a termination condition has been reached

Adaptive DE

exist different adaptative variations of the DE algorithm:

- **<u>DESAP :</u>** Not only dinamically adapts the parameter G.R. , but also the population size (Np).
- **<u>FADE :</u>** *Fuzzy Adaptive Differential Evolution* : Uses fuzzy logic to adapt the control parameters F, G.R. for the mutation and recombination operations.
- **<u>SaDE :</u>** (Self-Adaptive Differential Evolution) : The “F” and “G.R.” factors are independently generated according to a normal distribution.
- **<u>SaNSDE:</u>** generates mutation agents in the same method as SaDE exce that the mutation rate is generated according to either a normal or a Cauchy distribution.
- **<u>JDE :</u>** Adapts the control parameters “F” and “G.R.”associated with each agent.
- **<u>JADE :</u>** Adapts the control parameters “F” and “G.R.”associated with each agent and also, use a historical matrix to record the discarded agents like the BSA algorithm.

Adaptive BSA

ABSA : The probabilities of crossover and mutation are varied depending on the fitness values of the solutions
to refine the convergence performance.

Adaptive PSO

CLPSO : uses the best position of each agent to update a agent's velocity.

There are many more variants of the DE and PSO, only has tried to give an example of the existing variants

Acknowledgements

The supports from the Ministry of EU Affairs – Turkish National Agency under the Project Code: 2015-1-TR01-KA203-021342 are acknowledged.

References

- Cagnina, L. & Esquivel, S.C. 18/10/2013, *Particle swarm optimization para un problema de optimización combinatoria*, Red de Universidades con Carreras en Informática (RedUNCI), X Congreso Argentino de Ciencias de la Computación.
- Cagnina, L., Esquivel, S., Gallard R. "CEC2004: Proceedings of the 2004 Congress on Evolutionary Computation, Vols 1 and 2
- VIII CONGRESO COLOMBIANO DE METODOS NUMERICOS: Simulación en Ciencias y Aplicaciones Industriales 8CCMN – 2011, Agosto. 10-12, 2011, Medellín, Colombia ©2011 Universidad EAFIT
- Clerc, M. 2005, *Particle Swarm Optimization,* 1st edn, Import Diff.
- González-González, A., Etxeberria-Agiriano, I., Zulueta, E., Oterino-Echavarri, F. & Lopez-Guede, J. 2014, "Pitch Based Wind Turbine Intelligent Speed Setpoint Adjustment Algorithms.<br /> ", *Energies,* vol. 7, pp. 3793-3809.
- Oterino, F., Zulueta, E., Ramos, J.A., Calvo, I. & Lopez Guede, J.M. 2013, "Application of Differential Evolution as nethod of pitch control settings in a wind turbine", *Renevable Energy an Power Quality (RE&PQJ),* vol. 11.
- Oterino, F., Zulueta, E., Ramos, J.A., Lopez Guede, J.M. & Larrañaga, J.M. 2012, "Particle swarm optimization based tuning for a small wind turbine pitch control", *International Journal on "Technical and Physical Problems of Engineering",* vol. 4, no. 4, pp. 164-170.
- Price, K. & Storm, R.M. 2005, *Differential Evoluton: A practical Approach to global optimitation,* 1st edn, Springer, Alemania.
- Yu wang, Bin Li, Thomas Weise et al, 2011. "Self-adaptive learning based particle swarm optimization", Information Sciences, V 181, issue 20, 15 October 2011, Pages 4515–4538 Special Issue on Interpretable Fuzzy Systems

- Cagnina, L. & Esquivel, S.C. 18/10/2013, *Particle swarm optimization para un problema de optimización combinatoria*, Red de Universidades con Carreras en Informática (RedUNCI), X Congreso Argentino de Ciencias de la Computación.
- Sosa Toranzo, C. & Leguizamón, G. 2013, *Evolución diferencial como factor de mutación dinámico*, XVIII Congreso Argentino de Ciencias de la Computación.
- Storm, R.M. & Price, K. 1997, "Differential Evolution - A simple and efficient heuristic for global optimization", *Journal of global optimization,* vol. 11, pp. 341-359.
- Civicioglu, Pinar(2013) "*Backtracking Search Optimization for numerical optimization problems ",* Applied Mathematics and Computation Volume: 219 Issue: 15 Pages: 8121-8144.
- Teo, J.: Exploring dynamic self-adaptive populations in differential evolution. Soft Computation: Fusion Found Methodol Appl. 10(8), 673–686 (2006).
- Liu, J., Lampinen, J.: Adaptive parameter control of differential evolution. In: Proc. of MENDEL 2002, 8th International Mendel Conference on Soft Computing, Brno, Czech Republic, June 2002, pp. 19–26 (2002).
- Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: Proc. of IEEE Congress on Evolutionary Computation, September 2005, vol. 2, pp. 1785–1791 (2005)
- Yang, Z., Tang, K., Yao, X.: Self-adaptive differential evolution with neighborhood search. In: Proc. of IEEE Congress on Evolutionary Computation, Hongkong, China (2008)
- Brest, J., Boskovic, B., Greiner, S., Zumer, V., Maucec, M.S.: Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Computation: Fusion Found Methodol Appl. 11(7) (2007)

- Haibin Duan and Qinan Luo, “*Adaptive Backtracking Search Algorithm for Induction Magnetometer Optimization”* IEEE TRANSACTIONS ON MAGNETICS, VOL. 50, NO. 12, DECEMBER 2014.
- Liang, J.J., Qin, A.K., Suganthan, P.N. and Baskar, S.: ‘Comprehensive learning particle swarm optimizer for global optimization of multimodal functions’, *IEEE Trans. Evol. Comput., 2006, 10, pp. 281-295.*
- Zhan, Z.-H., Zhang, J., Li, Y. and Shi, Y.-H.: ‘Orthogonal learning particle swarm optimization’, *IEEE Trans. Evol. Comput., 2011, 15, pp. 832-847.*