Sisteme cu Inteligenta Artificiala

Heuristic and Metaheuristic Optimization Techniques with Applications

Main topics

- Optimization and (meta)heuristics
- Heuristic optimization
- Metaheuristics and metaheuristic methods
- Applications of (meta)heuristic methods in power systems
- Conclusions

Optimization and (meta)heuristics

Optimization

… is a branch of mathematics and computational science that studies methods and techniques specially designed for finding the "best" solution of a given "optimization" problem.

Such problems aim to minimize or maximize one or more objective functions based on one or more dependent variables, which can take integer or real values, and subject to a set of equality or inequality constraints.

Traditional optimization methods

- Linear Programming
- Integer Programming
- Quadratic Programming
- Nonlinear Programming
- Stochastic Programming
- Dynamic Programming
- Combinatorial Optimization

Difficulties faced by traditional optimization methods

- Passing over local optimal solutions
- The risk of divergence
- Handling constraints
- Numerical difficulties related to computing first or second order derivatives

(Meta)Heuristic methods

- Heuristic and metaheuristic techniques were proposed in the early 70's.
- Unlike exact methods, (meta)heuristic methods have a simple and compact theoretical support, being often based on criteria of empirical nature.
- These issues are responsible for the absence of any guarantee for successfully identifying the optimal solution.

Heuristic optimization

What is a heuristic ?

- A heuristic is an alternative optimization methods able to determine not a perfectly accurate solution, but a set of good quality approximations to exact solution.
- Heuristics, were initially based essentially on experts' knowledge and experience and aimed to explore the search space in a particularly convenient way.

Main characteristics

- A heuristic is designed to provide better computational performance as compared to conventional optimization techniques, at the expense of lower accuracy.
- The 'rules of thumb" underlying a heuristic are often very specific to the problem under consideration.
- Heuristics use domain-specific representations.

Types of heuristics

- Uninformed or blind search strategies are applied with no information about the search space, other than the ability to distinguish between an intermediate-state and a goal-state.
- Informed search strategies use problem-specific knowledge, such as an evaluation function that assesses either the quality of each state in the search space, or the cost of moving from the current state to a goal-state.

Uninformed search strategies

(basically non-heuristic)

- Depth First Search
- Breadth First Search
- Uniform Cost Search

Informed search strategies

Best First Search

Among all possible states at one level, the algorithm chooses to expand the most "promising" one in terms of a specified rule.

Informed search strategies

Beam Search

BeS is defined based on BrFS, which is used to build the search tree. At each level, all new states are generated and the heuristic function is computed for each state that is inserted in a list ordered by heuristic function values. The list is of limited length - "beam width". This limits the memory requirements, but the compromise risks to pruning out the path to the goal-state.

Informed search strategies

A* search algorithm

The A* search algorithm uses a BeFS strategy, and a heuristic function that combines two metrics: the cost from the origin to the current state (or the cost-so-far) and an estimation of the cost from the current state to a goal-state (or the cost-to-goal).

Metaheuristics and metaheuristic methods

What are metaheuristics ?

- The term metaheuristic was proposed by Glover at mid-80s as a family of searching algorithms able to define a high level heuristic used to guide other heuristics for a better evolution in the search space.
- The most attractive feature of a metaheuristic is that its application requires no special knowledge on the optimization problem to be solved (see the concept of general problem solving model).

Types of metaheuristics

- Simulated annealing
- Tabu search
- Evolutionary computation techniques
- Artificial immune systems
- Memetic algorithms
- Particle swarm optimization
- Ant colony algorithm
- Differential evolution
- Harmony search
- Honey-bee colony optimization

etcetera

Simulated Annealing

- Data: initial approximation $\mathbf{X_0}$, initial temperature $\mathbf{T}$, number of iteration for a given temperature $\mathbf{nT}$.
- Optimal solution: $\mathbf{X_{best}} \leftarrow \mathbf{X_0}$.
- WHILE {stopping criterion not met}
  - $\mathbf{n} = 0$; $\mathbf{i} = 0$;
  - WHILE ($\mathbf{n} < \mathbf{nT}$) DO
    - choose a new approximation $\mathbf{Y}$
    - accept or reject the new approximation based on the Metropolis rule: $\mathbf{X_{i+1}} = G(\mathbf{X_i},\mathbf{Y},\mathbf{T})$
    - update optimal solution: if $\mathbf{X_{i+1}}$ is better than $\mathbf{X_{best}}$, then $\mathbf{X_{best}} \leftarrow \mathbf{X_{i+1}}$
    - next $\mathbf{n}$-iteration ($\mathbf{n} \leftarrow \mathbf{n}+1$)
  - update temperature $\mathbf{T}$
  - next $\mathbf{T}$-iteration ($\mathbf{i} \leftarrow \mathbf{i}+1$)

Tabu Search

- Data: length of the Tabu list $\mathbf{L_T}$, number of intermediate solutions $\mathbf{N}$.
- Initialization: approximation $\mathbf{X}$, Tabu list $\mathbf{TABU}=\{\mathbf{X}\}$.
- Optimal solution: $\mathbf{X_{best}} \leftarrow \mathbf{X}$.
- WHILE {stopping criterion not met}
  - prepare the Tabu list: if Length(**TABU**) = $\mathbf{L_T}$, then delete the oldest item from the list.
  - generate $\mathbf{N}$ new aproximations in the neighborhood of $\mathbf{X}$ and select the best candidate-solution $\mathbf{Y}$ which is not **TABU**.
  - update current approximation $\mathbf{X} \leftarrow \mathbf{Y}$ and add it in the Tabu list: Add(**TABU**,**X**).
  - update optimal solution : if $\mathbf{X}$ is better than $\mathbf{X_{best}}$, then $\mathbf{X_{best}} \leftarrow \mathbf{X}$

Evolution Strategy

- Data: number of parents $\boldsymbol{\mu}$ and offsprings $\boldsymbol{\lambda}$ ($\boldsymbol{\lambda} = \mathbf{k} \cdot \boldsymbol{\mu}$).
- Initialization: create initial population $\mathbf{P} = \{P_i\}$, $\mathbf{i}=1 \ldots \boldsymbol{\lambda}$, and initialize the best solution **Best** $\leftarrow$ **void**.
- WHILE {stopping criterion not met}
  - evaluate $\mathbf{P}$ and update the best solution, **Best**.
  - reproduction stage: select $\boldsymbol{\mu}$ fittest individuals from $\mathbf{P}$ and create parent-population, $\mathbf{R} = \{R_j\}$, $\mathbf{j}=1 \ldots \boldsymbol{\mu}$.
  - mutation stage: apply stochastic changes to parents and create $\mathbf{k} = \boldsymbol{\lambda} / \boldsymbol{\mu}$ offsprings for each parent:

    $$[\, \mathbf{R} = \{R_j\},\ \mathbf{j}=1\ldots\ \boldsymbol{\mu}\,] \rightarrow [\, \mathbf{Q} = \{Q_i\},\ \mathbf{i}=1\ldots\ \boldsymbol{\lambda}\,]$$
  - Evolution stage: replace the current population with the mutated one:

    $$[\, P_i = Q_i,\ \mathbf{i}=1\ \ldots\ \boldsymbol{\lambda}\,]$$

Genetic Algorithms

- Data: population size **N**, crossover rate η_c and mutation rate η_m
- Initialization: create initial population **P**={P_i}, **i**=1…N, and initialize the best solution **Best** ← **void**.
- WHILE {stopping criterion not met}
  - evaluate **P** and update the best solution **Best**.
  - initialize offspring population: **R** ← **void**.
  - create offsprings:
    FOR **k** = 1 TO **N** / 2 DO
    - selection stage: select parents Q_1 and Q_2 from **P**, based on fitness.
    - crossover stage: use crossover rate η_c and parents (Q_1;Q_2) to create offsprings (S_1; S_2).
    - mutation stage: use mutation rate η_m to apply stochastic changes to S_1 and S_2 and create mutated offsprings T_1 and T_2.
    - add T_1 and T_2 to offspring population:
      $R \leftarrow R \cup \{ T_1 \text{ and } T_2 \}$.
  - replace current population P with offspring population **R**: **P** ← **R**.
  - elitism: replace the poorest solution in **P** with the best solution in **Best**.

Differential Evolution

- Data: population size parents **N**, weighting factors **α, β**.
- Initialization: create initial population **P**={P_i}, **i**=1…**N**.
- Evaluate current population **P** and store the fittest individual as **Best**.
- WHILE {stopping criterion not met}
  FOR **i** = 1 TO **N** DO
  - select two different individuals X_{r1} and X_{r2}, other than X_i .
  - apply mutation: $X_i' = X_i + \alpha\cdot(X_{r1} - X_{r2})$.
  - apply crossover: $X_i'' = X_i + \beta\cdot(X_i - X_i')$.
  - evaluate X_i'' and replace X_i with X_i'' anytime when Fitness (X_i'') > Fitness (X_i).
  - update the best solution **Best**.

Immune Algorithms

- Data: population, clonal and replacement size **N, M**, N_{rep}.
- Initialization: create initial population **P**.
- Evaluate affinities for antibodies in current population **P**.
- WHILE {stopping criterion not met}
  - clonal selection: clone the first **M** antibodies from **P**, with highest affinity. Number of clones for an antibody is proportional to its affinity. Number of clones in the proliferation pool **Q** is **N**.
  - mutation: apply stochastic changes to clones from the proliferation pool **Q**, with mutation rate inversely proportional to their affinity.
  - replacement: evaluate affinities for mutated antibodies and replace the worst N_{rep} clones from population **Q** with randomly generated antibodies.
  - elitism: evaluate new created antibodies and replace the worst antibody from **Q** with the best one from **P**.
  - next generation: replace current population with the one from the proliferation pool: **P** ← **Q**.

Particle Swarm Optimization

- Data: population size **N**, personal-best weight $\boldsymbol{\alpha}$, local-best weight $\boldsymbol{\beta}$, global-best weight $\boldsymbol{\gamma}$, correction factor $\boldsymbol{\varepsilon}$.
- Initialization: create initial population **P**.
- WHILE {stopping criterion not met}
  - select the best solution from the current **P**: **Best**.
  - select global-best solution for all particles: $\mathbf{B^G}$.
  - apply swarming:
    FOR **i** = 1 TO **N** DO
    - select personal-best solution for particle $\mathbf{X_i}$: $\mathbf{B_i^P}$.
    - select local-best solution for particle $\mathbf{X_i}$: $\mathbf{B_i^L}$.
    - compute velocity for particle $\mathbf{X_i}$:
      FOR **j** = 1 TO **DIM** DO
      - generate correction coefficients: $\mathbf{a} = \boldsymbol{\alpha} \cdot \text{rand}()$; $\mathbf{b} = \boldsymbol{\beta} \cdot \text{rand}()$; $\mathbf{c} = \boldsymbol{\gamma} \cdot \text{rand}()$.
      - update velocity of particle $\mathbf{X_i}$ along dimension **j**:
        $$\mathbf{V_{ij}} = \mathbf{V_{ij}} + \mathbf{a} \cdot (\mathbf{B_{ij}^P} - \mathbf{x_{ij}}) + \mathbf{b} \cdot (\mathbf{B_{ij}^L} - \mathbf{x_{ij}}) + \mathbf{c} \cdot (\mathbf{B_j^G} - \mathbf{x_{ij}})$$
    - update position of particle $\mathbf{X_i}$: $\mathbf{X_i} = \mathbf{X_i} + \boldsymbol{\varepsilon} \cdot \mathbf{V_i}$

Ant Colony Optimization

- Data: population size **N**, set of components $\mathbf{C} = \{C_1, \dots, C_n\}$, evaporation rate **evap**.
- Initialization: amount of pheromones for each component $\mathbf{PH} = \{PH_1, \dots, PH_n\}$; best solution **Best**
- WHILE {stopping criterion not met}
  - initialize current population, **P** = **void**.
  - create current population of virtual solutions **P**:
    FOR **i** = 1 TO **N** DO
    - create feasible solution **S**.
    - update the best solution, **Best** ← **void**.
    - Add solution **S** to **P**: $\mathbf{P} \leftarrow \mathbf{P} \cup \mathbf{S}$
  - Apply evaporation:
    FOR **j** = 1 TO **n** DO
    $PH_j = PH_j \cdot (1 - \mathbf{evap})$
  - Update pheromones for each component:
    FOR **i** = 1 TO **N** DO
    FOR **j** = 1 TO **n** DO
    - if component C_i is part of solution P_j, then update pheromones for this component: $PH_j = PH_j + \text{Fitness}(P_j)$

Honey Bee Colony Optimization

- Data: size of populations: - drones ($\mathbf{N_D}$), broods ($\mathbf{N_B}$) and genetic pool ($\mathbf{N_P}$); initial queen's speed $\mathbf{S_{max}}$; crossover rate $\boldsymbol{\eta_c}$ and mutation rate $\boldsymbol{\eta_m}$
- Initialization: create initial population **Drones** with $\mathbf{N_D}$ individuals, and select the best drone as the **Queen**.
- WHILE {stopping criterion not met}
  - create the genetic pool: use **Drones** population and select $\mathbf{N_P}$ individuals using a Simulated Annealing-type acceptance rule, based on the queen's speed **S**, and gradually reduce **S**.
  - crossover: apply arithmetic crossover between **Queen** and successively selected drones from the genetic pool, until a population of $\mathbf{N_B}$ broods (offsprings) is created.
  - mutation: apply arithmetic mutation to randomly selected broods (offsprings).
  - update the **Queen**: if any brood is better than the **Queen**, update the **Queen**.
  - selection: use broods and, based on a selection criterion, create the new population of **Drones**.

Applications of (meta)heuristic methods in power systems

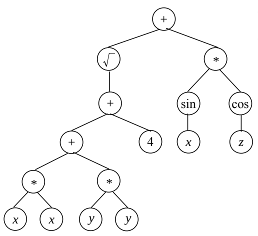
Types of applications

- Load assessment and profiling
- Network reconfiguration
- Reactive power planning
- System security analysis
- State estimation
- Distributed generation

any many others ….

Load assessment and profiling

Peak load estimation



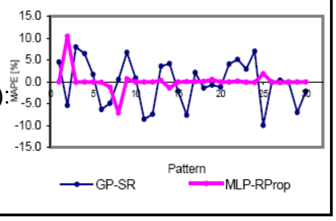
- **Dataset**: hourly load profiles
- **Problem**: find the approximant (analytical) that best fits the daily peak load using as input different combinations of data from the dataset.
- **Approach**: Genetic Programming & Symbolic Regression.
- **Solution-inputs**: peak loads form days d-7, d-6 and d-1 and the number of the reference day in the year.

Comparison of the GP&SR approach to an ANN-MLP model

Function approximation (GP&SR):

$$f1 = \cos(\cos(\sin x_3)) - \cos(x_3)$$
$$f2 = \cos(\cos(\cos x_3)) - \cos(x_3)$$
$$f = 2\cdot\cos(x_3) + x_1 - \log(x_2) - 2\cdot\log(\log(x_4)) - \log(f1) - 2\cdot\log(f2)$$

Estimation errors (GP&SR and ANN-MLP):



Load assessment and profiling

Load profiling (1)

- **Dataset**: hourly load profiles
- **Problem**: load profile clustering and typical load profiles generation.
- **Approach**: **Affinity Control inspired from Immune Algorithms and SOFM**.
- **Solution**: portfolio of typical load profiles.

Affinities between LPs and affinity degrees

Affinities between TLPs and metered LPs:

$$A_{i,j}^{T-T} = \sqrt{\left\| W_i - W_j \right\|} = \sqrt{\sum_{k=1}^{NK} \alpha_k \cdot \left(w_{i,k} - w_{j,k}\right)^2}$$

$$A_{i,p}^{T-M} = \sqrt{\left\| X_p - W_i \right\|} = \sqrt{\sum_{k=1}^{NK} \alpha_k \cdot \left(x_{p,k} - w_{i,k}\right)^2}$$

Affinity degrees:

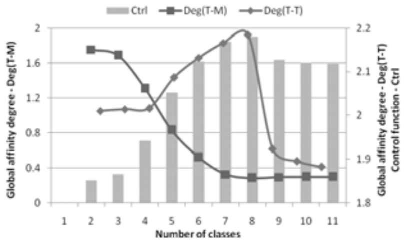
$$Deg^{T-T} = \frac{2}{NC \cdot (NC-1)} \cdot \sum_{i=1}^{NC-1} \sum_{j=1}^{NC} A_{i,j}^{T-T}$$

$$Deg^{T-M} = \frac{1}{NC} \cdot \sum_{i=1}^{NC} \sum_{p=1}^{NP} \left(A_{i,p}^{T-M} \Big/ \sum_{p=1}^{NP} \delta_{i,p} \right)$$

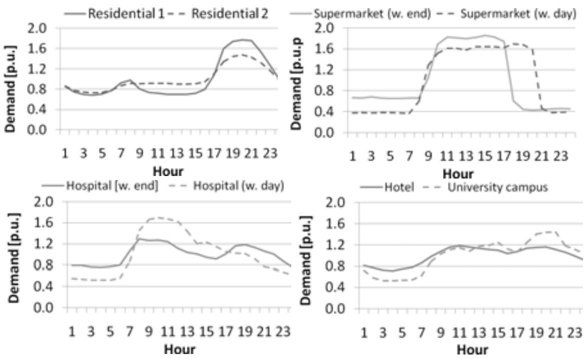
Control function:

$$Ctrl = \beta \cdot Deg^{T-T} - (1-\beta) \cdot Deg^{T-M}$$

Global affinity degrees and the control function during the growing phase



Results



Load assessment and profiling

Load profiling (2)

- **Dataset**: hourly load profiles
- **Problem**: load profile clustering and typical load profiles generation.
- **Approach**: Honey bee mating optimization algorithm.
- **Solution**: portfolio of typical load profiles.

Average distances between LP vs. Affinities

Average distances LP-TLP

$$\sigma_t = \frac{1}{NP_t \cdot NH} \cdot \sum_{i=1}^{NP} \| LP_i - TLP_t \| \quad Class(i) \equiv t \quad t = 1, \ldots, NT$$

$$\sigma = \frac{1}{NT} \cdot \sum_{t=1}^{NT} \sigma_t$$

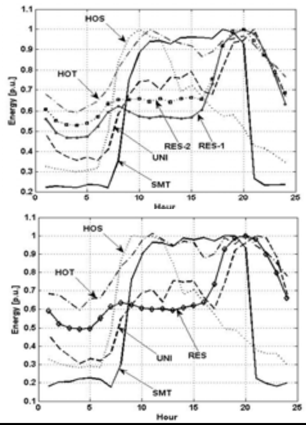
Average distances TLP-TLP

$$\rho_{s,t} = \frac{1}{NH} \cdot \| TLP_s - TLP_t \| \quad s,t = 1, \ldots, NT$$

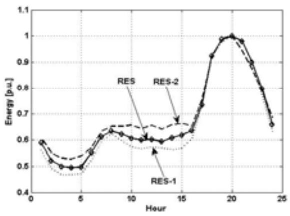
$$r_s = \frac{1}{NT-1} \cdot \sum_{t=1}^{NT} \rho_{s,t} \quad s = 1, \ldots, NT \qquad R = \frac{1}{NT} \cdot \sum_{s=1}^{NT} r_s$$

Fitness functions:

$$F_1 = 1/\sigma = 1/(\sum_{t=1}^{NT} \sigma_t / NT) \qquad F_2 = R - \sigma = \frac{1}{N_T} \cdot \sum_{s=1}^{NT} r_s - \frac{1}{N_T} \cdot \sum_{t=1}^{NT} \sigma_t$$



Results



Network reconfiguration

The network reconfiguration problem arises usually in distribution systems and aims at changing the network topology by altering the position and status of sectionalizing switches.

A complex combinatorial problem.

Network reconfiguration

Problem formulation (RPP included)

$$Min\left(\Delta P_{network}\right) = Min\left(\sum_{l=1}^{NL} \Delta P_l^{Line} + \sum_{t=1}^{NT} \Delta P_t^{Transf}\right)$$

subject to:

- Voltage constraints: $V_t^{\min} \le V_t \le V_t^{\max} \quad t = 1,\dots,NT$
- Line capacity constraints: $I_{l,1} \le I_{\max\,adm} \quad l = 1,\dots,NL$
- Capacitor constraints: $N_t \cdot \Delta Q_K \le Q_t^{\max} \quad t = 1,\dots,NT$; $\sum_{t=1}^{NT} N_t \le K_T$

Network reconfiguration (1)

- **Dataset**: network topology, network data, load data.
- **Problem**: determine the optimal combination of opened sectionalizing switches and the optimal RPP.
- **Approach**: Ant Colony Optimization.
- **Solution**: optimal configuration of the network that minimize power losses.

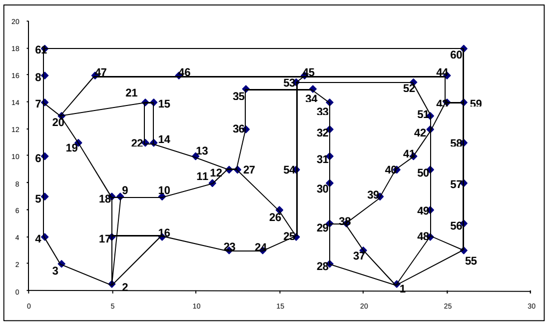
Network reconfiguration (1)

Solution representation:

- Reconfiguration problem:
  - A candidate solution will be represented by a vector consisting of the load sections that will be opened to obtain the radial configuration of the network.
- RPP problem:
  - A candidate solution will be a vector, which shows how many capacitor banks are placed in each node of the network.

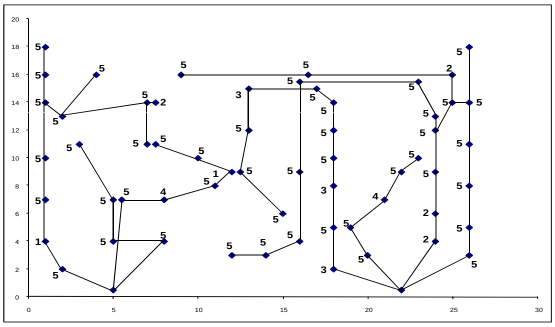
Network reconfiguration (1)

Test system:



Network reconfiguration (1)

Solution:



Network reconfiguration (2)

- **Dataset**: network topology, network data, load data.
- **Problem**: determine the optimal combination of opened sectionalizing switches and the optimal RPP.
- **Approach**: Particle Swarm Optimization (network reconfiguration) and Immune Algorithm (RPP).
- **Solution**: optimal configuration of the network that minimize power losses.

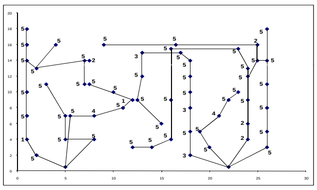
Network reconfiguration (1)

Solution representation:

A candidate solution encodes two m-tuples of indices. The first m-tuple indicates the feeders in the system where the sectionalizing switches should be opened, and the second m-tuple shows the line sections on each feeder where the switches must be turned off.

Network reconfiguration (2)

Example



- The same test system.
- The same optimal solution
- Comparable execution time.

Reactive Power Planning

Reactive power control:

- generator voltage control,
- transformer tap control and
- fixed or controllable VAR sources.

In distribution systems: Reactive Power Compensation (RPC) through power factor correction.

Reactive Power Planning

- **Dataset**: network topology, network data, load data.
- **Problem**: determine the optimal location and number of capacitors in the network.
- **Approach**: An enhanced Particle Swarm Optimization algorithm.
- **Solution**: optimal reactive power compensation that minimize power losses.

Reactive Power Planning

The enhanced Particle Swarm Optimization algorithm.

At each iteration a particle should try to mimic not only the best positions so far, but also other successful positions of itself and other particles in the current population.

$$v_i^{t+1} = c_0 \cdot v_i^t + c_1 \cdot \sum_{k=1}^{NT} r_{1,k} \cdot \left(B_{i,k}^L - X_i^t\right) + \\ + c_2 \cdot \sum_{k=1}^{NT} r_{2,k} \cdot \left(B_k^G - X_i^t\right)$$

Table 1 – Values of objective functions after 10 runs of three searching algorithms.

Reactive Power Planning

Comparison with other 2 algorithms.

Run #	GA	IA	PSO
1	2.6421	2.6206	2.6161
2	2.6368	2.6206	2.6155
3	2.6376	2.6217	2.6153
4	2.6343	2.6173	2.6126
5	2.6279	2.6196	2.6136
6	2.6280	2.6198	2.6126
7	2.6302	2.6209	2.6071
8	2.6275	2.6197	2.6209
9	2.6400	2.6203	2.6128
10	2.6416	2.6215	2.6120
Mean value	2.6346	2.6202	2.6139

System security analysis

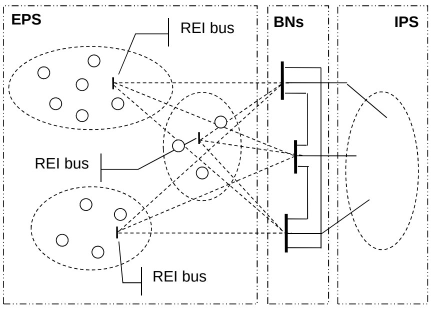
System equivalents are a good solution to simplify the on-line analysis of present day wide-area power systems used in assessing system security.

System security analysis

- **Dataset**: network topology, network data, load data.
- **Problem**: determine the optimal structure of the REI equivalent.
- **Approach**: Genetic Algorithm.
- **Solution**: How to group nodes in the REI equivalent ?

System security analysis

REI equivalent optimization



System security analysis

Solution representation (IEEE 57 test system):

Genes	2	3	1	3	2	2	2	3	1	2	4	1	4	2	3
Bus #	2	3	4	5	6	8	9	10	11	12	13	14	16	17	58

REI node # 1: group (4, 11, 14)

REI node # 2: group (2, 6, 8, 9, 12, 17)

REI node # 3: group (3, 5, 10, 58)

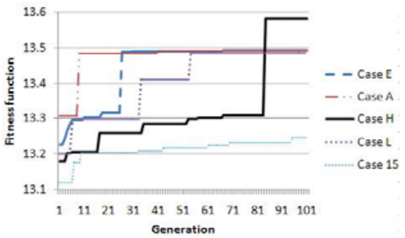
REI node # 4: group (13, 16)

Objective function:

$$FO=\frac{1}{NC\cdot NI}\cdot\sum_{i=1}^{NI}\sum_{j=1}^{NC}\frac{\left|U_{i,j}^{ref}-U_{i,j}^{eq}\right|}{U_{i,j}^{ref}}\cdot 100$$

System security analysis

Results (IEEE 57 test system):



Case	# REI nodes
A	2
E	3
H	4
L	5
15	15

Optimal solution:

REI node # 1: group (4, 11, 14)
REI node # 2: group (2, 6, 8, 9, 12, 17)
REI node # 3: group (3, 5, 10, 58)
REI node # 4: group (13, 16)

State estimation

- New applications for SE are largely based on Synchronized Phasor Measurement Units (SPMU).
- A hybrid solution which uses both traditional and SPMU is better in terms of costs.

State estimation

- **Dataset**: network topology, network data, SCADA measurements, PMU measurements.
- **Problem**: determine the optimal location of PMU measurements
- **Approach**: Genetic Algorithm.
- **Solution**: Buses where PMU measurements must be located.

State estimation

Solution representation (IEEE 14 test system):

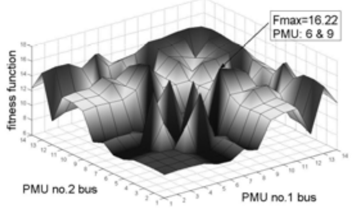
A chromosome consists of M blocks, each one associated to a PMU. A block describes the binary code of the bus where the PMU is placed.

A possible solution for a 2 PMUs placement-problem may use bus number 6 and 9, and the chromosome describing this solution will have the structure:

0	1	1	0	1	0	0	1

State estimation

Error surface for the case of 2 PMUs



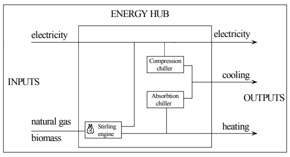
Applications 33/36

Distributed generation

Present day distribution systems are facing deep changing that transforms traditional design for passive operation into new concepts centered on distributed generation (DG) and a more active role of end-users.

Applications 34/36

Optimal design of multiple energy hubs

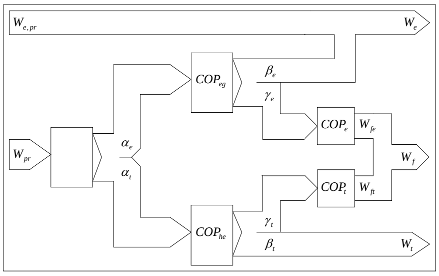


- **Dataset**: inputs and outputs of the energy hub, output loads, parameters of conversion units.
- **Problem**: determine the optimal input structure to supply the output loads.
- **Approach**: Genetic Algorithm.
- **Solution**: Inputs of the energy hub.

Applications 35/36

Optimal design of multiple energy hubs

Solution representation



A chromosome with 3 real coded genes:

α_e	β_e	β_t

Applications 36/36

Optimal design of multiple energy hubs

The objective function:

$$F_{obj} = k \cdot \left(W_{pr} + W_{e,pr}\right) + \left(1-k\right) \cdot dW_{pr}$$

where:

$F_1 = W_{pr} + W_{e,pr}$ - the primary energy function

$F_2 = dW_{pr}$ - error term equal to the difference between the primary energy computed on two independent ways

k - weighting coefficient

Conclusions

Conclusions 1/1

Conclusions

During the last two decades numerous (meta)heuristic approaches have been devised and developed to solve complex optimization problems.

Their success is due largely to their most important features:

- simplicity,
- the need of minimal additional knowledge on the optimization problem and
- a highly numerical robustness of algorithms.

END
of
Heuristic and Metaheuristic Techniques with Applications