ARTIFICIAL BEE COLONY ALGORITHM FOR POWER PLANT OPTIMIZATION

Friedrich Biegler-König
Department of Applied Mathematics
University of Applied Sciences
D-33615 Bielefeld
E-Mail: Friedrich.biegler-koenig@fh-bielefeld.de

KEY WORDS

Artificial Bee Colony Algorithm, Evolutionary Methods, Neural Networks, Mixed Integer Optimization, Power Plant Optimization

ABSTRACT

The mathematical model of a fleet of power plants can be optimized with respect to energy production. This involves the solution of a mixed integer problem. Traditionally these problems are solved by linearization of the continuous and non-linear parts and subsequent application of a Simplex-type algorithm. In order to handle nonlinearities, so-called "biomimetic" optimization algorithms can be applied. As an example, we are proposing an approach to first model power plant blocks with fast Neural Networks. Afterwards, we optimize the operation of multi-block power plants over a period of time using an Artificial Bee Colony Algorithm.

1. INTRODUCTION

Increasing prices for fossil fuels and the establishment of CO_2 credits to European markets have intensified the attempts to ensure energy efficient operation of existing power plants.

One of the well-known optimization issues where mathematical modelling has proven to be indispensable is the unit commitment in power generation planning. It deals with the scheduling of start-up/shut-down decisions and operation levels for a fleet of power generation units such that variable costs are minimal, or revenues are maximal. In this optimization, temporal constraints such as time dependency of energy prices at the power exchange, limited availability of certain plants, costs for start-up or shut-down, limits for total emission or fuel consumption within a given period of time should be taken into account. Therefore, the optimization has to be carried out within a certain time window with sufficient time resolution.

Because of the involvement of decision variables (power plant on or off), and the fact that every unit has a non-vanishing lower limit of energy that can be generated, the above-defined problem involves mixed integer programming. The well understood approach to this problem is to use the methods of mixed integer linear programming (Schultz 2003) in this context. It is widely used and allows for optimization with complex temporal constraints with reasonable CPU time consumption.

Recent studies (Biegler-König and Deeskow 2005, Deeskow et. al. 2005, Nolle 2007) have shown that a nonlinear approach has the advantage of easily taking into account the current state of each of the plant components and thus opens additional potential for optimization. It overcomes the generally high CPU time consumption of nonlinear modelling based on closed analytical models by meta-modelling the power plant using neural networks. However, this approach was restricted to a one-point-in-time optimization only.

In this contribution the approach presented in Biegler-König and Deeskow 2005 is extended to consider a model which is discrete in time and covers 24 hours of energy production with complex constraints. The resulting large mixed integer problem is treated by employing biomimetic optimization methods, in this case Simulated Annealing and Artificial Bees Colony algorithm.

2. APPROACHES TO MIXED INTEGER OPTIMIZATION

The common way to optimization in unit commitment uses a linear model of power generation and applies the simplex method of linear programming together with a Branch-and-Bound approach to solve the integer linear problem.

The simplex method is an iterative method to solve linear problems with constraints. Within a finite number of steps, it reaches the solution or proves infeasibility of the problem. The algorithm cannot be applied if some of the variables are restricted to integers. In that case methods are available which apply to a more or less general class of problems.

One of these methods is Branch-and-Bound, which consists of three phases. In the branching phase, the feasible region is partitioned, in the bounding phase linear

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

simplex is used to obtain upper and lower bounds of the optimal objective values and in the coordination phase rules are applied for eliminating parts of the feasible region from further consideration.

Unfortunately, the convenience of using the well-known mathematics of mixed integer linear programming comes to the expense of losing details in the modelling. This may have an impact on the potential for optimization.

A different approach is the use of so-called "Biomimetic Methods", i.e. methods which emulate the optimization behaviour of biological or physical systems. Well-known among these are Genetic Algorithms, Simulated Annealing (SA), or Artificial Bee Colony (ABC) algorithms (Pham et. al. 2006).

All these algorithms are iterative, based on heuristics, and have the advantage that they can easily deal with continuous variables as well as with discrete ones.

The number of iterations needed to produce good optimal states of the system is usually high. In most cases, however, it is impossible to prove that a state found is optimal.

The next section gives an introduction to the simulation approach for power plant blocks before SA and ABC algorithms are applied to a problem of the described type in sections 4 and 5.

3. FAST SIMULATION OF POWER PLANTS

We are considering a fleet of power plants with four blocks which are identical in construction (a simple case, but easily expandable). Each block can generate power in the range of 108 MW to 360 MW. A block cannot produce less than 108 MW. If a block is switched off, it can only be started up again at considerable costs.

A detailed model of a power plant block was built using Ebsilon, a programme developed by Sofbid (http://www.sofbid.com/ebsilon/, Brinkmann and Pawellek 2003 and 2004) in Zwingenberg, Germany. Ebsilon is a simulator specialized in power generating facilities. Ebsilon models can be extremely complex. After the model has been constructed, Ebsilon requires about 1 to 10 seconds on a PC to simulate a given situation for one block. This is by far not fast enough since many thousands of model evaluations are usually required for the considered algorithms.

A specific situation of the power plant block is determined by only a few input parameters. The most important parameter is the amount of energy which the block is to produce per time period.

The other parameters describe the external conditions and settings by the operational staff:

- Temperature of cooling water (between 0 and 30 °C).
- Air ratio in combustion chamber (between 1.1 and 1.4).
- Live steam temperature (between 510 and 540 °C).
- Hot reheat temperature (between 510 and 540 °C)
- Flue gas recirculation (between 0 and 50 kg/s).
- 8 more parameters specifying the amount of heat surface fouling.

As main result, Ebsilon supplies is the amount of fuel (in our case coal) required. This in turn determines the main part of the total costs.

As described in Biegler-König and Deeskow 2005, it is possible to replace the simulator Ebsilon by a much faster meta-model based on Neural Networks. The model-error of fitting a Neural Network to the Ebsilon model has the same magnitude as the model-error of an Ebsilon simulation.

The main advantage of the Neural Network model is its response time. It is about 10000 times faster than Ebsilon (response time: less than 0.0001 seconds).

4. SIMULATED ANNEALING

Simulated Annealing is based on the imitation of the cooling process of metals.

Here is a brief mathematical description of the algorithm:

Let $F(x_1, \ldots, x_N)$ be a target function in N variables. These variables can come from different sources: real numbers, binaries, integers, with or without lower and upper bounds. Additionally, these variables are subject to a set of restrictions. We assume that F possesses a global minimum whose position we want to determine.

Simulated Annealing is defined by the following iteration:

- For each component x_i of the vector x = $(x_1, \ldots, x_N)$ define a set of elementary steps C_i whose members change x_i .
- We start with an initial temperature of T_0, and define a final temperature T_E and a cooling factor $\alpha < 1$. We also have a start vector $x^0 = (x_1^0, \ldots, x_N^0)$, which satisfies all constraints. Let, for the beginning, $x^{min} = x^0$ and $F^{min} = F(x^0)$.
- In iteration no. i the following operations will be executed ($i = 1, \ldots$):
  1. Until a new candidate vector x^i is found which satisfies all constraints, choose a random component k of vector x^{i-1} and change it using a randomly chosen elementary step from C_i .
  2. Calculate $F(x^i)$.
  3. If $F(x^i) \leq F(x^{i-1})$, set $F^{min} = F(x^i)$. F^{min} is the new optimal value (so far).
  4. If $F(x^i) > F(x^{i-1})$, we accept x^i as the vector for the next iteration with the probability $p(T_i) =$

$exp((F(x^i)- F(x^{i-1}))/ T_i)$. As the algorithm sometimes accepts a worsening of the values, it is able to leave local minima.

5. If x^i was not accepted, set $x^i = x^{i-1}$.
6. Set $T_i = \alpha T_{i-1}$ and terminate the algorithm if $T_i < T_E$. The system has "cooled down".

The iteration process takes longer with larger T_0 and larger α. The probability of finding the true global minimum increases with the number of performed iteration steps.

Simulated Annealing is not population-based and easy to implement. There are many variations of this standard version of Simulated Annealing (see e.g. Nolle et. al. 2001). Possible changes are:

- A different probability function $p(T)$ can be chosen.
- Instead of a constant α a cooling function $\alpha(T)$ can be used.
- Elementary steps can be defined as being temperature dependent.

5. ARTIFICIAL BEE COLONY ALGORITHM

The Artificial Bee Colony (ABC) algorithm or Bees algorithm is a recent invention (Pham et. al. 2006). It emulates the behavior of a swarm of bees looking for food. A bee hive sends out a certain amount of "scouts" which look for promising food sources (e.g. flower patches). They return to the hive and communicate their findings by performing the "waggle dance". The scouts then go back to their food sources followed by other bees (more followers for better sources). Less favorable sources are abandoned; the corresponding scouts look elsewhere for more promising places.

The observation of the behavior of a bee swarm leads to a population-based swarm intelligence algorithm which can be used to optimize mathematical objective functions as defined above. In pseudo code, the basic form of the ABC algorithm looks like this:

1. Initialize population with random solutions.
2. Evaluate objective function for members of the population.
3. While stopping criteria are not met, build a new population:
4. Select best sites for neighborhood search ("Scouts").
5. Abandon other sites and send their bees to the neighborhood of selected sites.
6. Determine their objective function value.
7. For each neighborhood choose the bee with best objective function value.
8. Distribute all remaining bees randomly over the search space and evaluate their objective function value.
9. End While.

Step 8 will ensure that eventually all regions of the search space are considered. Population size and percentage of scouts are the control parameters of the algorithm. As in Simulated Annealing, elementary steps, i.e. a defined neighborhood of a candidate solution, must be defined.
In step 4, the scouts should all come from different neighborhoods in order to prevent premature convergence.
The distribution of bees to the scouts in step 5 may vary: sometimes all scouts get the same number of bees, sometimes the distribution is done proportionally to the quality of the scout site.

6. POWER PLANT OPTIMIZATION. A CASE STUDY

We will now use SA and ABC algorithms to optimize the performance of our fleet of four power plant blocks for the period of 24 hours.

We assume that every hour a different amount of energy must be produced and that these 24 values are given a day ahead. In reality, these estimates have a high accuracy. They contain information e.g. about the time of the year and weather forecast.

For each time t_i, $i = 1, \ldots, 24$ the production of our power plant with 4 blocks is described by the vector

$$x(t_i) = (E_1, E_2, E_3, E_4, b_1, b_2, b_3, b_4)^T.$$

$b_1, b_2, b_3,$ and b_4 are binary values and indicate whether or not a block is switched off. E_1, E_2, E_3, E_4 are the production rates of the blocks. In every hour of the day, they must add up to the given total production. We also have to consider the restrictions of minimal and maximal production rates: $108MW \leq E_i \leq 360MW$.

The objective function we want to minimize is the cost of energy production accumulated over the day. Since, for every hour in a day, we have a vector of eight variables, the total number of variables in our problem is 24x8 = 192. In our simple model, the total costs consist of the costs for coal and the start-up costs for blocks going on-line. Thus, the aim is not only to minimize the production costs in every single hour, but also to keep the number of switch-on-processes of power plant blocks low.

In order to implement the described algorithms we must still specify the elementary steps. Basically, two kinds of steps can be identified:

1. Change of production rate E_k for block no. k without altering the configuration of the power plant. These steps attempt to optimize a given configuration. This corresponds to solving the continuous part of the problem. In the beginning, a maximal step size S_{max} is

defined. S_{max} is multiplied by a random number between –1 and 1. This yields the current step size. The block number k is also chosen from the set of running blocks.
Afterwards, the restrictions are checked (minimal and maximal production rates, total energy production). If it is not possible to satisfy all boundary conditions, another elementary step must be chosen.
This kind of elementary step is applied with a probability of 90%.

2. Change of power plant configuration. In this case, the binary valued variables are changed. Let n be the number of running blocks. Possible changes of configuration are: Starting up an additional block (4-n alternatives), shutting down a running block (n alternatives), interchanging a running and a pausing block (m alternatives with m=3 if n=1 or n=3, m=4 if n=2, otherwise m=0).
This kind of elementary step is applied with a probability of 10%. Among these, steps are chosen according to the number of alternatives.
After interchanging two blocks, it is not necessary to adjust the production rates since, in our case, all blocks have the same limitations. Starting up or shutting down a block makes it necessary to readjust the production rates in order to assure the restrictions. If this is not possible, another elementary step must be chosen.

In addition to the elementary steps, the following control parameter settings have been used for SA:

- $T_0 = 10{,}000$.
- $T_E = 10^{-6}$.
- $\alpha = 0.99998$.
- Maximal number of iterations: $it_{max} = 1{,}000{,}000$.

The control parameters of ABC algorithm are:

- Population size = 50.
- Percentage of scouts: 10%
- Number of iterations: $it_{max} = 1{,}500$

With this choice of parameters, both algorithms found satisfactory solutions for the total costs over a day, although, of course, it cannot be verified that the solutions found represent global optima.

Example

For our example, we have used the described 4-block power plant of Elbistan (Turkey), a coal price of 30 Euros for a ton and a start-up price of 30,000 Euros for each block. For each hour of the day, a required energy production rate is given. Additionally, the values for heat surface fouling and cooling temperature for block 1 have been changed for the worse; the corresponding values of block 3 have been changed for the better. The parameters of blocks 2 and 4 are in between, the values of block 2 are slightly better than those of block 4.

Starting configuration and initial population have been chosen randomly, but satisfying all constraints. Random starting configurations are usually quite poor, since they contain many start-up processes (see Table 1). Tables 2 and 3 contain the results for SA and ABC algorithms, respectively, with the above settings of parameters.

Table 1: Initial Configuration

Uhrzeit	Vorgabe	Block 1	Block 2	Block 3	Block 4
00:00	530000	360000	170000	0	0
01:00	310000	0	0	0	310000
02:00	200000	200000	0	0	0
03:00	180000	0	0	0	180000
04:00	250000	250000	0	0	0
05:00	320000	0	0	0	320000
06:00	490000	360000	130000	0	0
07:00	650000	0	0	290000	360000
08:00	830000	360000	360000	110000	0
09:00	1030000	0	310000	360000	360000
10:00	1230000	360000	360000	360000	150000
11:00	1130000	282500	282500	282500	282500
12:00	1000000	360000	360000	280000	0
13:00	900000	0	180000	360000	360000
14:00	880000	360000	360000	160000	0
15:00	930000	0	210000	360000	360000
16:00	950000	360000	360000	230000	0
17:00	1000000	0	280000	360000	360000
18:00	1100000	275000	275000	275000	275000
19:00	1240000	160000	360000	360000	360000
20:00	1320000	360000	360000	360000	240000
21:00	1362300	282300	360000	360000	360000
22:00	1223000	360000	360000	360000	143000
23:00	830000	0	110000	360000	360000

Table 2: Solution Computed with SA

Uhrzeit	Vorgabe	Block 1	Block 2	Block 3	Block 4
00:00	530000	0	108565	211437	209998
01:00	310000	0	0	153338	156662
02:00	200000	0	0	199999	0
03:00	180000	0	0	180000	0
04:00	250000	0	126432	123566	0
05:00	320000	0	145880	174120	0
06:00	490000	0	241964	248036	0
07:00	650000	0	265798	272984	111218
08:00	830000	0	115082	359410	355507
09:00	1030000	0	359500	359540	310959
10:00	1230000	153591	359237	357749	359474
11:00	1130000	108104	339363	341302	341231
12:00	1000000	300353	110047	289906	299694
13:00	900000	110631	266104	267436	255828
14:00	880000	267615	111725	241880	258780
15:00	930000	116764	265973	273476	273786
16:00	950000	275284	129582	276409	268724
17:00	1000000	302995	293152	294674	109179
18:00	1100000	108162	329713	330481	331645
19:00	1240000	161862	359837	358664	359637
20:00	1320000	359020	241419	359825	359734
21:00	1362300	357927	359971	359908	284493
22:00	1223000	144060	359202	360000	360000
23:00	830000	0	357988	359923	112090

Table 3: Solution Computed with ABC Algorithm

Uhrzeit	Vorgabe	Block 1	Block 2	Block 3	Block 4
00:00	530000	0	170000	360000	0
01:00	310000	0	0	309999	0
02:00	200000	0	0	199999	0
03:00	180000	0	0	179998	0
04:00	250000	0	0	250000	0
05:00	320000	0	0	320000	0
06:00	490000	0	129999	360000	0
07:00	650000	0	290000	360000	0
08:00	830000	0	342045	359948	128007
09:00	1030000	0	310001	360000	359999
10:00	1230000	150002	359997	360000	360000
11:00	1130000	146206	263796	359999	360000
12:00	1000000	112163	284973	313019	289845
13:00	900000	154280	235359	351367	158995
14:00	880000	137467	222878	281802	237853
15:00	930000	237643	220202	323749	148406
16:00	950000	146816	251191	360000	191991
17:00	1000000	360000	153117	359999	126884
18:00	1100000	157037	273416	360000	309547
19:00	1240000	160001	360000	360000	359999
20:00	1320000	360000	360000	360000	239999
21:00	1362300	360000	282301	360000	359999
22:00	1223000	143001	360000	360000	359999
23:00	830000	0	338076	360000	131924

The first column of the tables contains the time of day, the second the required energy production in kW. Columns 3 to 6 show how the blocks share the production of this energy amount.

The ABC algorithm required about 10 million evaluations of the underlying neural network while SA took about half as many evaluations and, subsequently, about half the computation time. Increasing the number of iterations in SA does not improve the results. The time required for both methods on a normal PC stays well below 30 minutes, thus being always faster than reality.

Both algorithms provide qualitatively similar solutions. They run block 3, which has the highest efficiency, with full capacity. Block 1 is only used if necessary since its efficiency is worst.

SA started with a configuration which costs 1,818,778 euros (configuration of Table 1) for the day and lowered it to 1,297,099 euros (configuration of Table 2). The initial 20 start-up processes where reduced to 3.

ABC algorithm started with a population whose best member costs 1,438,616 euros a day and contains 7 start-up processes. The result configuration of Table 3 costs 1,281,726 euros and also contains 3 start-up processes.

The main weakness of SA is that it may lose good solutions and finally concentrate on less favourable ones. ABC always keeps the so far best solutions and tries to improve on them. This can clearly be seen by comparing the two tables.

The above behaviour of both algorithms was verified with different starting values.

7. CONCLUSION

This paper has discussed a concept for online optimization in power plant technology which combines different numeric state-of-the-art processes to construct a method customized for power-engineering requirements.

Neural Networks can be trained with high accuracy to reproduce the results of the computationally intensive thermodynamic simulation programs on the basis of physical fundamentals. As such, non-linear detailed models of the individual power plant units can be generated, whose response time lies within the range of milliseconds.

Simulated Annealing is a heuristic optimization algorithm, which appears to be suitable for solving the mixed-integer optimization problems arising in power plants and which, together with fast neural network models, allows an online solution for the non-linear optimization problems.

The Artificial Bee Colony algorithm, a population-based heuristic optimization algorithm, yields better results, even though it needs more computation time.

In an earlier study, this concept was successfully applied to a momentary optimization, which already resulted in substantial savings of costs and fuel. The present study demonstrates that a suitable extension of the objective function allows for optimization of a model which is discrete in time and considers constraints linking different time intervals. Thus, it is clearly shown that the described methods allow nonlinear mixed integer optimization of unit commitment in a fleet of power plants.

Because of the non-linear nature of the models, such approaches would be able to solve problems in the future, which so far either cannot be solved at all or only with difficulty using a traditional optimization on the basis of a linear modelling. Examples here are the consideration of the non-linear boundary conditions (temperatures in the steam piping) or a common optimization of the load distribution to multiple units and, at the same time, the mode of operation of the individual units.

REFERENCES

Biegler-König, F. and P. Deeskow. 2005. “Fast Simulation and Optimization with Neural Networks”. *Proc. 19th European Conference on Modelling and Simulation*, Riga.

Brinkmann, K. and R. Pawellek. 2003. “Ebsilon - Examples for the easier design and better operation of power plants”. *Proc. Conf. Energy Forum 2003*, Sv. Konstantin Varna.

Brinkmann, K. and R. Pawellek. 2004. “Optimierte Prozessführung von Kraftwerksblöcken mit Online-Werkzeugen: Betriebserfahrungen“. *Proc. VDI Tagung “Wissensbasiertes Betriebsmanagement senkt Kosten“*, Frimmersdorf.

Deeskow, P.; F. Biegler-König; L. Nolle. 2005. “Schnelle Optimierung von Kraftwerksparks mit Neuronalen Netzen“. *Proc. 6. VDI-Fachtagung Optimierung in der Energiewirtschaft*, Stuttgart.

Nolle, L. ”Non-linear Total Energy Optimisation of a Fleet of Power Plants”. 2007. *Applications and Innovations in Intelligent Systems XIV*.

Pham, D.T.; A. Ghanbarzadeh; E. Koç; S. Otri; S. Rahim; M. Zaidi.2006. “The Bees Algorithm – A Novel Tool for Complex Optimisation Problems”, *Proceedings of IPROMS Conference, pp. 454–461.*

Schultz, R. 2003. Course Material: “Integer Programming Applied to Power Systems' Generation and Operation Planning”. Preprint 557-2003, Institut für Mathematik, Gerhard-Mercator-Universität Duisburg.

L. Nolle, A. Goodyear, A. Hopgood, P. Picton, N. Braithwaite, On Step Width Adaptation in Simulated Annealing for Continuous Parameter Optimisation. *Proc. 7th Fuzzy Days*, Dortmund, 2001, 589-598.