Optimisation of Concentrating Solar Thermal Power Plants with Neural Networks *

Pascal Richter[1,2], Erika Ábrahám[1], and Gabriel Morin[2]

[1] Chair of Computer Science 2, RWTH Aachen University, Aachen, Germany
[2] Fraunhofer Institute for Solar Energy Systems, Freiburg, Germany

Abstract. The exploitation of solar power for energy supply is of increasing importance. While technical development mainly takes place in the engineering disciplines, computer science offers adequate techniques for simulation, optimisation and controller synthesis.
In this paper we describe a work from this interdisciplinary area. We introduce our tool for the optimisation of parameterised solar thermal power plants, and report on the employment of genetic algorithms and neural networks for parameter synthesis. Experimental results show the applicability of our approach.

Keywords: Optimization, Solar thermal power plants, Neural networks, Genetic algorithms

1 Introduction

The contribution of renewable energies to global energy supply has significantly increased over the past ten years. Completely new branches of industry have developed in the fields of solar, wind, and biomass energy. Among such technologies, *concentrating solar thermal power (CSP) plants* are a promising option for power generation in regions with high direct solar irradiation. The principle seems to be very simple: Large mirrors concentrate rays of sunlight to heat water and the emerging vapour powers a turbine to generate electricity (see Fig. 1).

In the early planning stage of commercial CSP plants, it is necessary to develope a conceptual *plant design* that fixes the *configuration* of the plant, such as the solar field size and the temperature and pressure levels in the water cycle. In the ideal case the design minimises the *levelised cost of electricity (LCOE)*, describing the costs per generated electricity unit, for the given project and site, and taking specific properties like solar conditions and cooling water availability into account.

In this paper, we describe our simulation-based techno-economical *optimisation tool* for the development of such project-specific plant concepts that are well-designed with respect to economic criteria. The optimisation tool uses adequate

* This work is based on the Fraunhofer ISE project "optisim", which was funded by the German Ministry of Environment (project number FKZ 0325045).

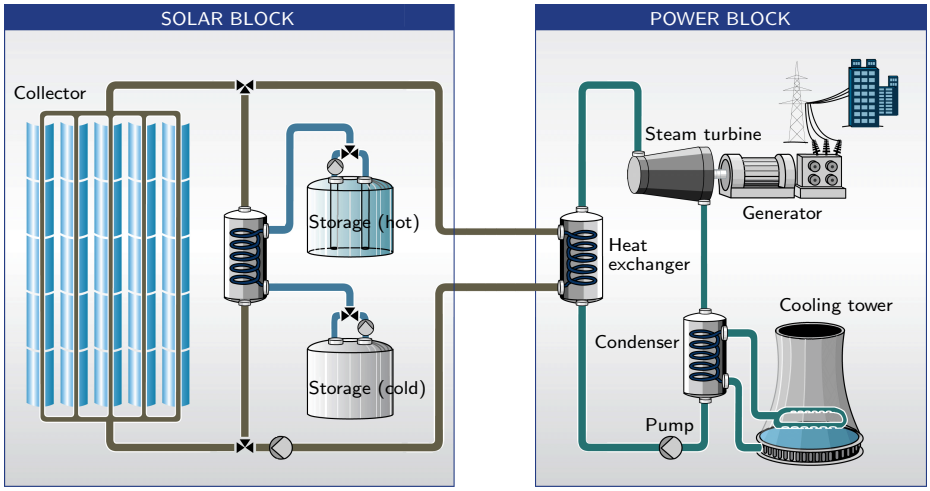

Fig. 1. Structure of a concentrating solar thermal power plant. In the solar block, large mirrors collect rays of sunlight and concentrate them on an absorber pipe. The concentrated heat of the sun is used to heat a transfer fluid, usually thermal oil. The hot fluid is either sent to the power block, where in a heat exchanger hot steam is generated from liquid water, or its energy is stored in a molten salt storage for later use after sun-set. In the power block the vapour streams through the turbine to power its blades so that the generator generates electricity.
Source: Solar Millennium 2009, own modifications

computer science techniques: The global optimum of a solar plant configuration is found using a genetic algorithm, whereas the thermodynamic-energetical procedures of a solar plant are described by an artificial neural network.

The physical behaviour of CSP plants is very complex, so their analytical optimisation is in practice not possible. Instead we use *genetic algorithms* [3] for the optimisation. The basic idea is very simple: Given an arbitrary set of CSP configurations, we simulate each of them energetically and economically to compute their LCOE average over a year. The LCOE serves as the objective function for the optimisation. We select the best configurations with minimal costs and combine them to get a new generation of configurations for which we repeat the procedure. By mutation this method avoids getting stuck in local minima.

In order to get reasonable LCOE values for a CSP configuration, we need to simulate the plant behaviour for each hour of a year. Without further improvements (see Section 4) the optimisation would take about 1000 days due to the time-consuming thermodynamical power block simulation using Thermoflex [12].

To reduce the calculation time we approximate the thermodynamical power block simulation by *bilinear interpolation*: For each considered configuration, instead of simulating each hour of a year we simulate only at an experimentally

determined number of interpolation points. In this way we are able to reduce the running time of the optimisation to 2 days.

To further reduce the running time, we train a *neural network* [5] to learn the required function of the power block behaviour, and replace the simulation by the trained neural network. This is the main contribution of this paper, whereby the computation time is reduced to about 2 hours (including training).

Related work There are several tools that simulate CSP plants (e.g. Thermoflex [12]). However, these tools are not able to optimise the configurations of CSP plants. Morin [10] connected in his PhD thesis Thermoflex (for power block simulation) with the solar block simulation tool ColSim [14, 10] and with an optimisation algorithm using genetic algorithms and bilinear interpolation (but no neural networks). To our knowledge this is the only work on global optimisation of CSP plant designs, including power block design.

There are also papers on combining genetic algorithms [3] with neural networks [5]. The NNGA approach applies neural networks to find advantageous initial sets for the genetic algorithm (see, e.g., [6]). In reverse, the so-called GANN approach uses genetic algorithms to set the parameters of the neural network. A broad variety of problems has been investigated by different GANN approaches, such as face recognition [4], Boolean function learning and robot control [9], classification of the normality of the thyroid gland [11], color recipe prediction [2], and many more. In contrast to the above approaches we use in our work neural networks to generate the input data for the genetic algorithms.

The rest of the paper is structured as follows. Section 2 describes the simulation of CSP plants. Section 3 is devoted to the optimisation using genetic algorithms. We use bilinear interpolation and employ neural networks to speed up the optimisation in Section 4. After presenting experimental results in Section 5, we conclude the paper in Section 6 with an outlook on future work.

2 The Simulation of CSP Plants

The aim of optimising concentrating solar thermal power plants is to generate electricity as cheaply as possible. The cost-efficiency of a power plant is generally specified by the so-called levelised cost of electricity (LCOE), which describes the costs per generated electricity unit (e.g. in Eurocent per kWh).

We consider up to 20 *design parameters* of a CSP plant. Examples for such parameters are solar field size, storage capacity, condenser size, distance between collector rows, as well as pressure and temperature levels. We use $\boldsymbol{p}_{design}$ to denote the design parameters of the CSP plant (see Fig. 1 for the CSP plant structure). Our goal is to find a configuration of these parameters that yields a minimal LCOE.

For a fixed configuration of the CSP plant the LCOE must be calculated under consideration of the seasonal and daily variations of its *site parameters*: The direct normal solar irradiance (DNI) has an influence on the collected thermal power in the solar block and the ambient temperature influences the cooling

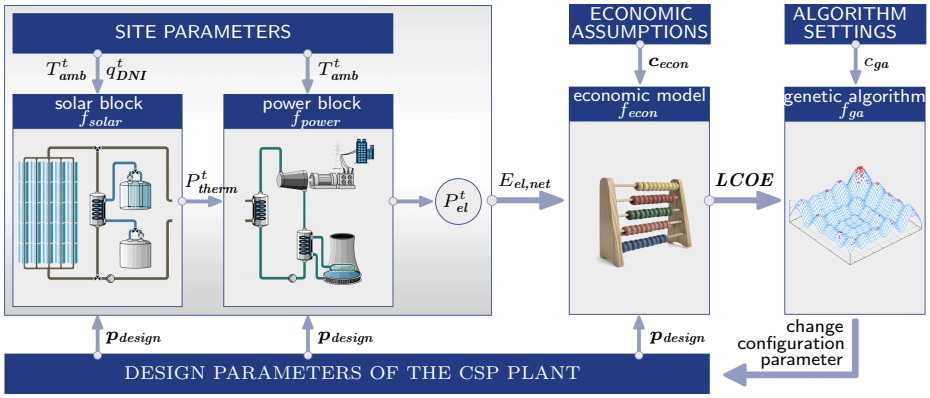

Fig. 2. General structure of the optimisation procedure. Given a configuration fixing the design parameters of the CSP plant, we use the solar and power block models to simulate the CSP behaviour and compute the generated electrical energy for each hour of the year, under consideration of the site parameters. Summing up the generated electrical energy for each hour of the year gives the total energy amount, used as input to the economic model to compute the LCOE under some economic assumptions. The LCOE is the basis for the genetic algorithm to evaluate the configuration of the CSP plant and to create new generations of configurations.

section of the power block[3]. Therefore, the model for computing the LCOE is based on hourly time resolution over one year. We use q^t_{DNI} and T^t_{amb} to denote the DNI and the ambient temperature for the tth hour of the year.

The LCOE of a CSP plant for a given configuration is computed in two steps: Firstly, we determine the electrical net energy $E_{el,net}$ generated over a year using an *energetic model* of the CSP plant. Secondly, this value is used by the *economic plant model* to compute the LCOE under consideration of economic assumptions.

2.1 The Energetic Model of a CSP Plant

Physically the electrical energy is defined as the time-integral of the electrical power: $E = \int_t P_{\mathrm{el}}(t)\ \mathrm{d}t$. The electrical net energy generated over a year by a CSP plant for a given configuration is approximated with numerical rectangle method as the sum of the electrical power generated during each hour t of a year: $E_{el,net} = \sum_{t=1}^{8760} P^t_{el}$.

To compute the electrical power P^t_{el} for the tth hour of a year, we first compute with the help of the *solar block model* the thermal power P^t_{therm} gained by the solar block. This value serves as an input to the *power block model* that determines the generated power P^t_{el} (see Fig. 2).

The *solar block model* provides a function f_{solar} to calculate the thermal power $P^t_{therm} = f_{solar}(T^t_{amb}, q^t_{DNI}, \boldsymbol{p}_{design})$ based on the available direct normal

[3] The hotter the ambient air, the less efficient the power plant

solar irradiance q^t_{DNI} and the configuration $\boldsymbol{p}_{design}$ of the solar block. The calculation first determines the optical collector performance and then subtracts heat loss and thermal inertia effects (when heating up or cooling down). We use the ColSim tool [14, 10] for these calculations. Depending on the operation strategy, either heat is stored or hot fluid is sent to the power block.

The efficiency of converting thermal energy into electrical energy in the power block depends on the thermal power P^t_{therm}, the ambient temperature T^t_{amb} (influencing the cooling section), and the configuration $\boldsymbol{p}_{design}$ of the plant. The *power block model* provides a function f_{power} to specify for each hour t the electrical energy $P^t_{el} = f_{power}(P^t_{therm}, T^t_{amb}, \boldsymbol{p}_{design})$. The power block receives a thermal energy flow from the solar field and/or the storage and converts it first into mechanical and then into electrical energy. Mass and energy balances are computed for each component (e.g., steam turbine, condenser and pumps). We use the Thermoflex [12] tool for these computations.

The total electric net energy $E_{el,net} = \sum_{t=1}^{8760} P^t_{el}$ serves as an input to the economic model.

2.2 The Economic Model of a CSP Plant

The *economic plant model* specifies a function f_{econ} to calculate the LCOE. The total investment costs for the solar block and the power block are computes depending on economic assumptions $\boldsymbol{c}_{econ}$ (e.g. investment costs of collectors, interest rate, etc.) and the configuration $\boldsymbol{p}_{design}$. The investments occuring at the initial project phase are distributed over the lifetime of a plant using the annuity factor. On top of the investment-related annuity the running costs occuring in the phase of operation of the plant need to be added. The annual running costs consist of: Staff for operation and maintenance of the plant (e.g. mirror washing), water, spare parts and plant insurance. These economic assumptions are included in $\boldsymbol{c}_{econ}$. The levelised cost of electricity $LCOE = f_{econ}(\boldsymbol{c}_{econ}, \boldsymbol{p}_{design}, P^{total}_{el})$ equals the quotient of the annual costs and the electrical energy generated over a year. The ColSim tool also supports these computations.

3 Use of Genetic Algorithms to Optimise Solar Plants

As described above the techno-economical model can be used to compute the LCOE of a configuration. However, the number of possible configurations grows exponentially in the number of parameters. To compute the LCOE for every configuration in the search space is not realisable in practice. Hence, we need an efficient heuristic approach to approximate such a multi-dimensional optimisation problem.

Genetic algorithms, a special type of evolutionary algorithms, are well-suited for this purpose because they do not require knowledge about the problem structure. Furthermore, they can easily handle discontinuities, which is of crucial importance here since technically unrealisable configurations have to be sorted out.

Another kind of discontinuities comes from several technological solutions (e.g. integer number of collector loops).

A set of initial *individuals* (in our case configurations) widely spread over the whole search area form an initial *generation*. Analogous to biological evolution, genetic algorithms select the best individuals (in our case the configurations with the smallest LCOE's) from the current generation and use features such as selection, recombination and mutation to produce a new generation. Iterative application of this procedure leads to individuals close to the optimal solution.

Our optimisation tool embeds such a genetic algorithm. The implementation is based on the free C++ library GALib [13]. We treat undesirable or contradictory configurations by penalisation: they get assigned a very high LCOE and are thus discriminated in the subsequent selection process.

On average, the genetic algorithm needs about 25 iterations with 40 configurations per iteration to get close to the global minimum.

4 Improvements of the Optimisation

The running time of the optimisation based on a genetic algorithm is mainly determined by the simulation of the power block (see Section 2.1) that calculates the generated electrical energy. A typical characteristic diagram for the function f_{power} of the power block model is shown on Fig. 3. The computation of the function values must consider complex physical processes, and is therefore very time-consuming. For a single configuration, the computation of the electrical energy P_{el}^{i} generated during a certain hour i of a year needs about 10 seconds[4] on a standard computer. That means, it takes about a day to compute the electrical energy P_{el}^{total} generated over a year. The genetic algorithm considers about 1000 configurations until it gets close to the optimum. Thus the optimisation would need around 1000 days without further improvements.

For applications we need to reduce the computation time. Below we present two improvements to speed up the optimisation. The first approach involves *bilinear interpolation*, whereas the second one employs *artificial neural networks*.

4.1 Bilinear Interpolation

Assume a power block configuration $\boldsymbol{p}_{power}$ is given. Instead of computing the generated electrical energy $P_{el}^{t} = f_{power}(P_{therm}^{t}, T_{amb}^{t}, \boldsymbol{p}_{power})$ for each hour t of a year we compute it only for the grid points of a two-dimensional grid in the space of thermal power and ambient temperature. We use these grid points together with their computed function values to interpolate the function f_{power} for the given configuration, i.e., to get approximations for P_{el}^{t} for each hour of the year. We use bilinear interpolation for this purpose, which performs linear interpolation first in one dimension and then in the other dimension.

Experiments have shown that it is sufficient to compute the values of f_{power} for a 4×4 grid, i.e., to simualate a configuration, the modified power block model

[4] See Section 5 for more details.

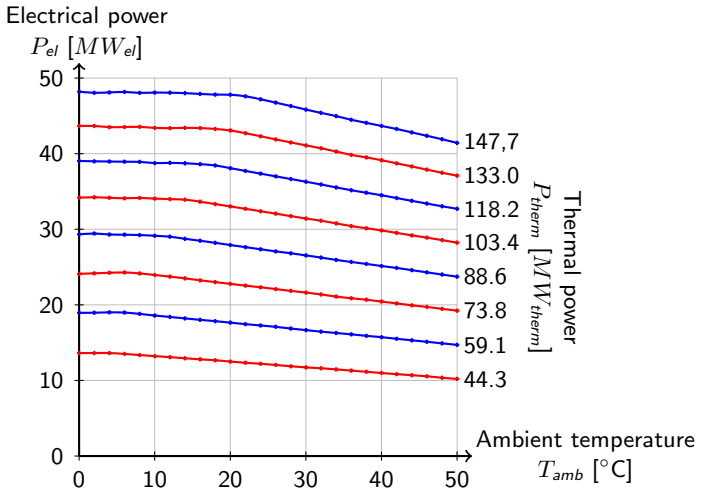

Fig. 3. Typical characteristic diagram of the function f_{power} for a fixed power block configuration specifying the electrical power for each thermal power and ambient temperature value pairs.

needs only 16 simulations (10 seconds each) instead of 8760 simulations. As the genetic algorithm needs about 1000 configurations to get close to the optimum, we have an approximate running time of about 2 days.

4.2 Artificial Neural Networks

To further reduce the computation time, we could think of applying linear interpolation in the dimension of configurations as we did for the environmental state. However, the behaviour in this dimension is highly non-linear, which leads to an unacceptable effect on simulation accuracy.

For this purpose we use *neural networks* [5] instead of linear interpolation. Neural networks are able to learn how to approximate functions without knowing the function itself. The learning process is based on a training set consisting of points from the domain of the function as input and corresponding function values as output. We use neural networks to learn the behaviour f_{power} of the power block model, i.e., to determine the generated electrical energy for a given configuration and for some thermal power and ambient temperature values.

There exists a wide range of different neural networks. We use *multilayer perceptron*, a network consisting of different neuron layers. As shown in Fig. 4, there is a flow of information through a number of hidden layers from the input layer which receives the input, to the output layer which defines the net's output. Fig. 5 shows the general scheme of the neurons in the layers. Each neuron weights its inputs and combines them to a net input on the basis of a transfer function (usually the sum of the weighted inputs). The activation function determines the

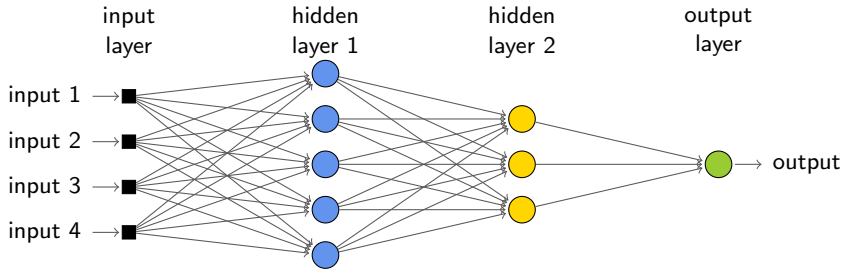

Fig. 4. Topology of an example multilayer neural network with two hidden layers.

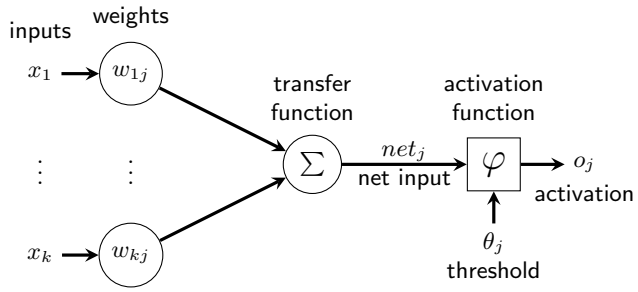

Fig. 5. Scheme of an artificial neuron j.

activation output under consideration of a threshold value. During the learning process the number of hidden layers and the number of neurons in the layers do not change, but each neuron adopts the weights of its inputs.

We train the multilayer perceptron during the optimisation as follows: Before the network is trained we apply bilinear interpolation as before. The results are used as training set for the network, configuration and environmental state are used as input and the interpolated values as output. Experiments show that it is sufficient to train with the first 20 configurations. After the network has been trained we use the neural network approach instead of the bilinear interpolation. For each of the 20 configurations we need 16 simulations for the bilinear interpolation, leading to a total simulation time of about 2 hours. The remaining computation times (bilinear interpolation, training, etc.) are insignificant compared to the simulation times. The quality of the results is comparable to the results of the approach using only bilinear approximation.

5 Experimental Results

We use the *Flood* tool [7] to define, train, and use multilayer perceptrons with two hidden layers. The networks can be configured by fixing different network parameters. The configuration influences the network's output, which again influ-

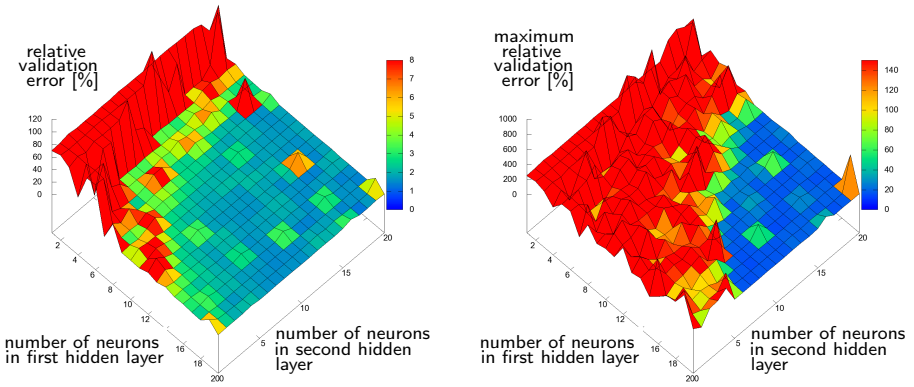

Fig. 6. The relative (left) and maximum relative (right) error for multilayer perceptrons, all having 2 hidden layers but a different number of neurons in the hidden layers.

ences the quality of the optimisation result. To attain good results it is therefore of high importance to find an appropriate configuration.

We determined a well-suited network configuration experimentally: For each parameter we trained networks with different parameter values and compared the quality of the outputs. As a measure for the quality of the training we used the relative error between the target and the predicted values for a test set with 12000 data points.

During training [1] a network tries to adapt its free parameters (edge weights and neuron thresholds) such that the difference between the target values and the network output for a set of training points gets smaller. There are different training algorithms using different objective error functions. The best predictions were received by application of the conjugate gradient algorithm as training algorithm and the regularised Minkowski error with regularisation weight $s = 0.1$.

As optimal parameters we determined as optimal parameters the sum of the weighted inputs as the transfer function and the hyperbolic tangent as the activation function for the neurons.

We fixed the above parameter values and varied the number of neurons in the hidden layers. Fig. 6 presents the relative validation errors. The best results were gained with 15 neurons in the first and 17 neurons in the second hidden layer with a relative error of 1.2%.

With these settings of a neural network, an annual simulation of a CSP plant predicted the LCOE with a relative error of 0.67% and a maximum error of 1.3%. The effects for the optimisation of using a neural network should be determined in future work. It is expected, that the approximated optimum determined by using a neural network should be close to the optimum found by a time-consuming simulation-based optimisation.

6 Conclusion and Outlook

We described and applied an approach to optimise concentrating solar thermal power plants by determining economically optimal design parameters. The combination of simulation, genetic algorithms, bilinear interpolation and neural networks allowed us to reduce the calculation time of the optimisation procedure by around 90% compared to an approach without neural networks.

Neural networks were used here to detect complex thermodynamic analogies between different plant designs. The achieved accuracy for prediction of the LCOE is a relative error of less than 1%. In this paper simple multilayer perceptrons were used. In future work here is room for improvements, e.g. using recurrent neural networks [8], which would need fewer parameters to optimise and due to their feedback, they are likely to suit the problem more.

References

1. C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.
2. J.M. Bishop, M.J. Bushnell, A. Usher, and S. Westland. Genetic optimisation of neural network architectures for colour recipe prediction. In *International Joint Conference on Neural Networks and Genetic Algorithms*, pages 719–725, 1993.
3. D.E. Goldberg et al. *Genetic algorithms in search, optimization, and machine learning.* Addison-wesley Reading Menlo Park, 1989.
4. P. Hancock and L. Smith. GANNET: Genetic design of a neural net for face recognition. *Parallel problem solving from nature*, pages 292–296, 1991.
5. J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554, 1982.
6. E. Liau and D. Schmitt-Landsiedel. Automatic worst case pattern generation using neural networks & genetic algorithm for estimation of switching noise on power supply lines in cmos circuits. *European Test Workshop, IEEE*, pages 105–110, 2003.
7. R. Lopez. Flood: An open source neural networks C++ library. www.cimne.com/flood, Universitat Politècnica de Catalunya, Barcelona, 2008.
8. D.P. Mandic and J.A. Chambers. *Recurrent neural networks for prediction: Learning algorithms, architectures and stability.* Wiley, 2001.
9. V. Maniezzo. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks*, 5(1), 1994.
10. G. Morin. *Techno-economic design optimization of solar thermal power plants.* PhD thesis, Technische Universität Braunschweig, 2010.
11. W. Schimann, M. Joost, and R. Werner. Application of genetic algorithms to the construction of topologies for multilayer perceptrons. In *International Joint Conference on Neural Networks and Genetic Algorithms*, pages 675–682, 1993.
12. Software Thermoflex: software developed and distributed by Thermoflow Inc. http://www.thermoflow.com/.
13. M. Wall. GAlib: A C++ library of genetic algorithm components. http://lancet.mit.edu/ga/, 1996.
14. C. Wittwer. *ColSim Simulation von Regelungssystemen in aktiven Solarthermischen Anlagen.* PhD thesis, Universität Karlsruhe, 1998.