3rd European supercritical CO2 Conference
September 19-20, 2019, Paris, France

2019-SCO2.eu-123

MACHINE LEARNING BASED DESIGN OF A SUPERCRITICAL CO_2 CONCENTRATING SOLAR POWER PLANT

Tahar Nabil*
EDF R&D China Center
Beijing, China
Email: tahar-t.nabil@edf.fr

Yann Le Moullec
EDF R&D China Center
Beijing, China

Adrien Le Coz
EDF R&D China Center
Beijing, China

ABSTRACT

Replacing steam water by supercritical CO_2 is a promising track for increasing the thermal efficiency of concentrating solar power plants, likely to facilitate the development of this industry. Adopting this new working fluid requires first a careful optimal design of the power cycle. In this work, we propose a novel, fully data-driven approach to address this task. Unlike state-of-the-art methods, which rely on some expert knowledge and an intrinsically limited search space, our method freely explores the solution space in an unconstrained and efficient manner. We revisit cycle design and formulate this problem as a machine learning task consisting in training a statistical generator. Starting from a limited number of random power cycle layouts, we are able to train a model, namely a recurrent neural network, from which new layouts can be sampled. These new configurations are thermodynamically valid, i.e. they do correspond to cycles from which a thermal efficiency can be computed. Besides, we show how to transform the model to bias the samples towards regions of both high thermal efficiency and net shaft power. The final model is efficient, with at least 86% of the newly created layouts outperforming a target specification of 30% thermal efficiency and 10 MW net shaft power. Numerical experiments finally prove that the method successfully generates a large pool of good cycles that compare favourably with standard expert designs, suggesting thus that machine learning can challenge expert knowledge in process synthesis problems.

INTRODUCTION

Improving the thermal-to-electric conversion efficiency of thermal power plants is an active field of research, because of the pressing need for reducing the carbon dioxide (CO_2) emissions released by the power industry. Whereas most thermal plants, whether they are nuclear, fossil or concentrating solar power (CSP) plants, are based on a steam cycle, closed-loop Brayton cycles using supercritical carbon dioxide (SCO_2) as a working fluid are now being investigated [1–4]. Compared to steam, SCO_2 is thought to bring cost and environmental impact reduction as a result of an increased efficiency, a more compact turbomachinery, and a simpler configuration of the power block. These desirable properties stem from the high-density and strongly non-linear behaviour of carbon dioxide near its easily reachable critical point, at 30.98°C and 73.8bar. In particular, the compression work for CO_2 cycles near the critical point is greatly reduced compared to other working fluids [1]. Looking more specifically at CSP cycles, advantageously replacing steam by SCO_2 could thus further foster the development of this emerging industry. Indeed, while CSP coupled with thermal energy storage is able to handle the intrinsic variability of the solar radiation by decoupling the solar-to-thermal and thermal-to-electric conversions, its cost remains high - typically from 60 to around 100 $/MW in the near future according to the International Renewable Energy Agency (IRENA). By improving the thermal-to-electric efficiency of the power cycle, SCO_2 could help lowering the cost of CSP. However, while the techno-economic performances of the steam cycle for CSP are well understood, SCO_2 cycles remain largely theoretical objects whose layout and parameter optimization should be further examined. How then to synthesize a highly efficient SCO_2 power cycle for CSP application?

This task, called process design, is a complex parametric optimization problem. Expert methods were first developed: some cycle layouts are selected using domain knowledge, and the free parameters of each layout are then optimized with respect to some criteria, for instance energy efficiency [5], exergy efficiency [6] or cost and performance trade-off [7]. Typical layouts are the regenerative, recompression, intercooling, pre-heating cycles, etc. [8]. This strategy fails nevertheless to design power cycles automatically, and do not enable the comparison of many configurations, such that more relevant ones could be missed.

Optimization-based approaches have thus emerged, to extend expert designs. They usually rely on a superstructure which contains a combinatorial number of layout alternatives. For instance, the superstructure in [9] involves 1152 layouts of a SCO_2 coal-

* corresponding author

DOI: 10.17185/duepublico/48885

fired power plant. Superstructure optimization is a mixed-integer nonlinear programming problem, where the best cycle configuration and its corresponding best design parameters are found simultaneously [9, 10]. Although the solver converges towards an optimal configuration *within* the superstructure, the optimization-based approach suffers from the same limitation as the expert design. Even if larger, the search space remains intrinsically limited by the current state-of-the-art knowledge since the superstructure is drawn by an expert. Potentially relevant solutions *outside* the superstructure are therefore also left out. Besides, defining a comprehensive superstructure is a complex error-prone task.

Consequently, optimization-based yet superstructure-free synthesis approaches are currently searched for in order to better explore the solution space. In this regard, see for instance the use of evolutionary algorithms applied to a population of power cycles in [11, 12]. Such algorithms usually iterate through three main steps:

1. generate new power cycles;
2. evaluate their fitness (e.g. their thermal efficiency);
3. learn from the previous steps to search for better cycles.

Step 1 focuses on proposing new power cycle layouts. In [11–13], this is achieved by carefully choosing the mutation operator of the evolutionary algorithm. Step 2 requires solving a generally nonlinear optimization problem for every fitness computation. Finally, step 3 may take several forms, e.g. selection rules in evolutionary algorithms [11–13]. A distinguishing feature of the new methodology is thus its *genericity*, with little to no application-specific knowledge. Another feature is the *data-driven* approach, by evaluating the fitness of successive sets of possible candidates, until the algorithm converges towards a pool of good power cycles with high fitness. Because these already existing data-driven contributions require either further validation on the complex power plant design problem [13] or a non-negligible expert knowledge for designing the mutation operator [11], we identify the need for exploring new fully data-driven approaches going beyond evolutionary algorithms.

Study objectives

Our main purpose is to propose a new data-driven process synthesis method able to challenge expert designs. The method should freely and efficiently explore the search space, in order to produce a large set of *good* power cycles and look for solutions experts could have missed. The recent advances of deep learning [14] suggest to formulate this machine learning problem as a generative statistical task. This means that, given a predefined set of power cycles, we aim at estimating a statistical model from which we can sample new power cycles, close but distinct from the initial set. As a first step, we investigate whether a state-of-the-art deep learning model, namely the recurrent neural network, is able to fulfill this task by generating new layouts that are all thermodynamically valid, i.e. that correspond to actual power cycles whose thermal efficiency can be assessed (regardless of its actual value). The second objective is then to derive a statistical generator that not only produces new, thermodynamically valid, layouts, but also concentrates specifically on highly efficient designs. We will use transfer learning, i.e. adjust the parameters of the statistical model trained during the first step, to derive this specialized model.

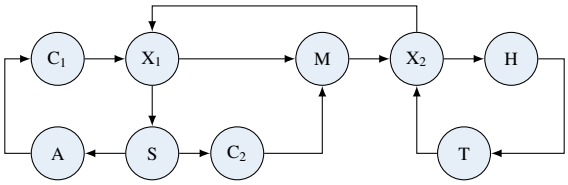

Figure 1: Graph representing the recompression SCO_2 power cycle. T: turbine, C: compressor, A: cooler, H: heater, X: heat exchanger with cold (resp. hot) stream flowing horizontally (resp. vertically), S: splitter, M: mixer.

We focus here on SCO_2 CSP power cycles, with heat transferred from molten salt (60 wt% $NaNO_3$ and 40 wt% KNO_3), and solar tower technology. Yet, the methodology applies equivalently to any other power cycle. Before describing in more details the models and results obtained in this work, the next section introduces the data representation and available dataset.

DATASET

1. Representing power cycles

Graph representation

In this paper, a supercritical carbon dioxide power cycle is represented by a directed graph. The components of the cycle are the elementary units that may be part of any power cycle, namely a heater (molten salt-CO_2 heat exchanger), a turbine, a compressor, a cooler, a CO_2-CO_2 heat exchanger, and 2-branch splitters and mixers. They are converted into the vertices of the graph, whereas CO_2 flowing from component A to component B is equivalent to a directed edge between the corresponding nodes. Without further input, the graph only informs about the topology (layout) of the cycle, and not about its thermal state. As an example, see the recompression cycle depicted in Figure 1.

String representation

We introduce furthermore a string representation of a power cycle, and state that a power cycle may be uniquely represented by a *word*, i.e. an ordered sequence of letters. Consider indeed the alphabet $\mathscr{A} = \{$`T, C, H, A, a, b, c, d, 1, 2, 3, 4, 5, -1, -2, -3, -4, -5`$\}$, where script-size letters `a, b`, etc. denote semi-heat-exchangers, a positive integer denotes a mixer and its opposite value the corresponding splitter, while `T, C, H, A` are as in Figure 1. The alphabet $\mathscr{A}$ is given in its (arbitrarily defined) lexicographic order. A power cycle graph is then translated into a word as follows:

1. start from an empty string, select the unit ranked first in lexicographic order, i.e. a turbine, and append the corresponding letter to the string. The word reads thus `T`. In case of multiple

DOI: 10.17185/duepublico/48885

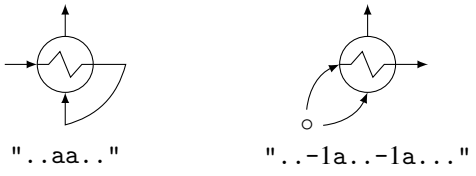

Figure 2: Examples of thermodynamically infeasible (left, fluid heating itself) or unsuitable (right, shared inlet conditions) heat exchanger designs, and corresponding words with spelling mistakes.

turbines, select the one whose successor appears first in $\mathscr{A}$. *Example:* for a cycle including two turbines, one followed by a heat exchanger, the other by a heater, choose then the latter one, since `H` comes before `a` in $\mathscr{A}$.

2. "follow the flow": move to the unique successor of the turbine, append its associated letter to the word, and iterate until the successor of the current component is the initial turbine.

A few special rules are added to guarantee the uniqueness of the word. (i) Every heat exchanger in the cycle appears twice in the word, for it has two sides. These two sides are denoted by the same letter, e.g. `a`, meaning that they belong to the same exchanger - it is not necessary to distinguish between cold and hot. (ii) 2-branch splitters and mixers always come in pairs, one pair should be denoted by a single index in absolute value (e.g. a pair (`-1, 1`)). (iii) The first encountered heat exchanger should be denoted `a`, the second `b`, etc. i.e. the numbering respects the standard lexicographic order. Similarly for pairs of splitters/mixers. (iv) Assume the current component is a splitter. Because it has two successors instead of one, the "follow the flow" rule is complemented as follows:

- append its letter (say e.g. `-1`) to the word;
- explore the first branch in lexicographic order, follow the flow and augment the word, until the associated mixer `1` is met;
- if the mixer has been seen before, i.e. `1` appears now twice in the word, then continue the exploration forward, i.e. move to the unique successor of the mixer `1`;
- otherwise, return to the root splitter `-1`, augment the word by following the flow in the other branch until mixer `1` is met. Mixer `1` appears now twice in the word, so the exploration continues forward from `1`.

A splitter appears thus exactly once in the word, its corresponding mixer exactly twice. *Example:* the word equivalent to the recompression cycle in Figure 1 is `Tab-1C1ACb1aH`.

The benefits of using words lie in the uniqueness and compactness of the representation. Because of the specific properties of a power cycle, the lexicographic order defined on $\mathscr{A}$ and these simple spelling rules, there is indeed a one-to-one bijection between graphs and words. Hence, the finite alphabet $\mathscr{A}$ and the rules defined in this section form a *language* for describing power cycle layouts. This formal grammar is built in analogy to the SMILES format for representing molecular structures [15].

Table 1: Supercritical CO2 CSP plant design constraints.

Symbol - description	Value
T_{max} - maximum operating temperature	560 °C
T_{min} - minimum operating temperature	35 °C
P_{max} - maximum operating pressure	250 bar
dp_c - pressure drop (cooler)	0.5 bar
dp_{hx} - pressure drop (heater, exchanger)	1 bar
ΔT - temperature pinch (heat exchanger)	5 K
η_c - isentropic efficiency (compressor)	0.89
η_t - isentropic efficiency (turbine)	0.93
$\dot{m}$ - molten salt mass flow	80 kg/s
T_1 - molten salt minimum temperature	290 °C
T_2 - molten salt maximum temperature	565 °C

2. Available data

In order to assume the least possible expert knowledge in the course of process design, the initial database consists of a set of random cycles. More precisely, building on the graph representation, each cycle is obtained firstly by creating a closed graph containing the minimal set of units, i.e. turbine, cooler, compressor and heater. Then, a random number of random components are inserted in the cycle at random locations, with equal probabilities. Finally, graphs that are thermodynamically infeasible are discarded. To do so, they are first converted into words. Then, a set of hand-designed rules are established in order to link invalid power cycles to invalid sequences of letters. Indeed, the link between graph structure and thermodynamic feasibility can easily be spotted by using the equivalent string representation. *Example:* Figure 2 shows some sequences of letters that yield unsuitable cycles. For instance, words including the pattern `"..aa.."` are rejected, because they correspond to cycles having an exchanger whose outlet stream of the first half is used as input to the second half. Similarly the pattern `"..-1a..-1a..."` is unsuitable, since it means that the exchanger `a` has two inlet streams at the exact same thermodynamic conditions (coming from the same source, the splitter `-1`). These two rules, and additional ones omitted here for sake of brevity, are enough to assess the validity of a word.

Fitness evaluation

A labeled dataset is obtained by maximizing a user-defined fitness function, chosen here to be the product $\mathscr{P} \times \eta$ of the net shaft power output $\mathscr{P} = (|P_t| - |P_c|)$ in MW and the thermal efficiency $\eta = \mathscr{P}/Q_{abs}$ in %, where P_t is the power generated by all the turbines of the cycle, P_c the power consumed by all the compressors and Q_{abs} the total heat transferred to the working fluid in molten salt/CO_2 heat exchangers. Each power cycle configuration is subjected to the same set of design constraints, representative of the expected technology advancement for developing a 10 MW SCO_2 CSP plant with solar salt (60 wt% $NaNO_3$ and 40 wt% KNO_3) tower technology, and summarized in Table 1.

The free parameters are the target temperatures of the heaters

DOI: 10.17185/duepublico/48885

and coolers, the pressure ratios of the turbomachinery as well as the mass fraction of the splitters. The static equations leading to η may be found in any thermodynamics textbook, e.g. [16].

This problem is implemented in Python, with thermodynamic fluid properties computed by the CoolProp library [17]. NetworkX [18] is used to handle graph data, with the thermodynamic state of each component stored as node attribute. The nonlinear optimization task is solved by a Particle Swarm Optimizer (PSO, [19]) with 100 randomly initialized particles. The best particle after convergence of the PSO is then used as initialization of a simplex algorithm (Nelder-Mead algorithm, [20]).

METHODS

Given the dataset described in the previous section, we are now ready to formulate our machine learning task. Starting from a collection $\mathscr{D}_0$ of power cycle layouts, the first step towards data-driven process design is to learn a statistical model able to sample from the unknown distribution underlying $\mathscr{D}_0$, i.e. able to produce new power cycle layouts, representative but distinct from the cycles in $\mathscr{D}_0$. Using the string representation, the model should thus generate new valid words. By *valid word*, we mean a sequence of letters in the alphabet $\mathscr{A}$ that can be converted into a power cycle from which a thermal efficiency actually can be computed. *Example:* `Tab-1C1ACb1aH` (recompression cycle) and `TaACaH` (regenerative Brayton cycle) are valid sequences, but `TAH` (no compressor `C` in the cycle) and `TaaACH` (infeasible heat exchanger) are not. The purpose is therefore to learn "the language of power cycles", we refer to this task as the *language, or spelling, learning*.

The second task consists then into specializing the statistical model such that it generates only power cycles with certain desired properties, for instance a high thermal efficiency.

1. Recurrent neural networks

The statistical generative model is chosen to be a recurrent neural network (RNN), a class of neural networks that allows feedback to handle sequential data [21]. RNNs are used e.g. in natural language processing, to automatically generate meaningful sequences of words, i.e. sentences [22]. Power cycles are also sequential data, being ordered sequences of units represented by letters in $\mathscr{A}$. The inputs of our RNN will be sequences of fixed-size vectors, where each unit is converted into a unique vector by adopting one-hot encoding: to the i^{th} letter of $\mathscr{A}$ corresponds a vector of size $|\mathscr{A}|$, whose entries are 0's, except the i^{th} one set to 1. A power cycle is thus a sequence of one-hot vectors. RNNs are attractive models since they allow for persistency of the information, i.e. memory, and can deal with correlations between data points close in the sequence. Besides, they can handle inputs of variable dimension, i.e. power cycles with a variable number of units.

Assuming that it operates over vectors ${x^{(t)}}_{t=1}^{N}$, with index t refering to the order of appearance in a sequence, a RNN is a N-layer neural network, with the concept of memory of the network translated by introducing the notion of a cell. At a given step t, the $cell_t$ is the result of the previous $cell_{t-1}$ and the current input $x^{(t)}$. The content of $cell_t$ determines then both the output at the current step and the next state of $cell_{t+1}$, as shown in Figure 3. Let vector h_t denote the hidden state of $cell_t$, h_t encapsulates the memory of the network. We have $h_t = \sigma(Ux^{(t)} + Wh_{t-1})$, with σ a nonlinear function, e.g. the sigmoid $\sigma : \mathbb{R} \ni x \mapsto 1/(1+\exp(-x)) \in (0,1)$. The output at step t is $o^{(t)} = f(VH_t)$, with f a given function. For instance, if $f =$ softmax, the vector $o^{(t)}$ is then a probability distribution across $\mathscr{A}$ for the next item $X^{(t+1)}$ in the sequence. The weight matrices U, V, W are shared across time: the same task is performed at each step, only with different inputs.

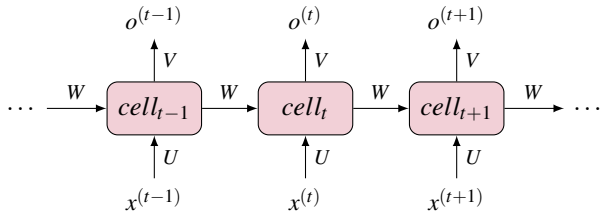

Figure 3: Unfolded recurrent neural network representation.

Long short-term memory A major shortcoming of conventional RNN is their poor ability to catch long-term dependencies in the data, failing to account for close relations between $x^{(t)}$ and $x^{(t')}$ for $t' \gg t$. Long short-term memory (LSTM) networks, a subclass of RNNs, were thus introduced to address this issue [23]. They share the structure depicted in Figure 3, except that the design of the cell is modified: LSTM cells have fours layers, against one for RNNs. These four layers interact in order to remove or add information to the cell state. See [23] for further information on the detailed structure of LSTMs.

Generative net Consider a RNN whose output $o^{(t)}$ is the probability distribution of the next input $x^{(t+1)}$: $o^{(t)} = P(x^{(t+1)}|x^{(1)},\dots,x^{(t)})$. For instance, here, the RNN computes the probability of the i^{th} component of a cycle given the first $i-1$ units. Suppose, by convention, that any word starts with a `GO` token and ends with an `EOS` (end of sequence) token. Then, once the RNN is trained, it can be used to generate new sequences: start with a `GO` token, sample the next element in the sequence from the output $P(x^{(t+1)}|x^{(1)},\dots,x^{(t)}) = P(x^{(2)}|x^{(1)})$, and iterate until an `EOS` token is sampled. Hence, sequences of variable length may be generated.

2. Language learning from few data

In practice, training RNNs requires large amounts of diversified data, which we do not have. We proceed thus to data augmentation and suggest to use RNNs alternatively in training and generative modes in order to enlarge an initial set $\mathscr{D}_0$ of random cycles.

1. Set index $k = 0$.
2. Use the cycles in $\mathscr{D}_k$ to train a RNN model.
3. Use the model learned in 2. to generate new cycles, filter out the invalid cycles, append the remaining ones to $\mathscr{D}_k$, forming a new, augmented, dataset $\mathscr{D}_{k+1}$.

DOI: 10.17185/duepublico/48885

4. $k \leftarrow k+1$, return to 2. until a maximum number of iterations is reached.

For the procedure to be relevant, we use the rules described in the Dataset section to automatically filter out invalid words. The rules are applied at step 3 of the procedure, and not during step 2: the trained model is merely told what to imitate (the input data), not what to avoid. Observe also that the RNN trained from $\mathscr{D}_{k+1}$ is not a finetuned version of the RNN trained from $\mathscr{D}_k$, since its parameters are randomly initialized. Each model is trained from a database of increasing size, such that after a few iterations, the RNN should be able to mostly generate valid power cycles.

3. Finetuning

For the purpose of power cycle design, the actual machine learning task we aim at is to generate *highly efficient* cycles, not merely power cycles. Here, we want power cycles with fitness $\mathscr{P} \times \eta > \delta$, for a predefined threshold δ. Because the number of cycles in $\mathscr{D}_0$ meeting this criterion is very small, training a RNN directly from this subset would produce an overfitted model. Instead, the desired model is achieved by a process called transfer learning, or finetuning. Firstly, a RNN is trained to generate thermodynamically valid layouts, regardless of their fitness value. This language model, denoted $\mathscr{M}_1$, is obtained following the methodology described in the previous section. $\mathscr{M}_1$ is then re-trained on a smaller subset of cycles sharing the property $\mathscr{P} \times \eta > \delta$, yielding a new model $\mathscr{M}_2$ with the same RNN architecture but different parameters. $\mathscr{M}_1$ learns thus general features of power cycles; this knowledge is stored in its parameters after training. Although $\mathscr{M}_2$ is trained on a different dataset, its parameters are initialized to the values obtained by $\mathscr{M}_1$. In this manner, knowledge (about how to draw a thermodynamically valid layout) is transferred to the second model (specialized in generating highly efficient power cycles).

In the end, new layouts are sampled from the finetuned model. We emphasize the fact that this deep learning method does not provide any guarantee regarding its convergence towards the "best" layout. Its interest lies rather in its ability to create large sets of data, hopefully meaningful for the process engineer. We say thus that the exploration of the space of power cycles is efficient if the new cycles do have their fitness above the predefined threshold δ.

RESULTS AND DISCUSSION

1. Experimental settings

The purpose of the numerical experiments is two fold. Starting from a small database of power cycles layouts, the first step consists in generating virtually infinitely many new, distinct, power cycles - regardless of their thermal efficiency. The second task is more specific: we want to generate power cycles whose layout guarantees a minimum level of thermal performance of the cycle. The first task is achieved by (i) training a RNN on the initial, random, database, (ii) using this model in generative mode, to produce new cycles (iii) append valid new cycles to the initial database, and (iv) iterate until the training set has sufficiently grown and the learned model has reached a good success rate in generation mode. In this case study, we train the RNN to predict the next letter given the first part of a word (sequence-to-one model). Hence, to every power cycle w of length n in the training set correspond $n-1$ pairs of input-output data for the RNN. These pairs are formed by the one-hot sequences of the first i letters of w as input, and the one-hot representation of the $i+1^{th}$ letter as output, for $i = 1, \ldots, n-1$.

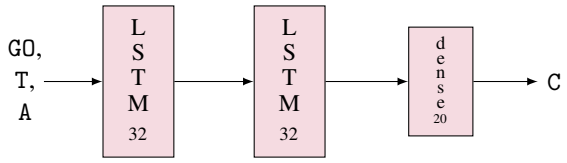

Figure 4: Structure of the RNN model for generating new power cycles. In training mode, the model learns to predict the component (compressor `C`) that comes next given the beginning of the cycle (`GO` token, turbine `T`, cooler `A`).

The architecture of the RNN model is displayed in Figure 4. It has 3 stacked layers, two LSTM layers, each with a hidden state vector of size 32, and a dense output layer with softmax activation. The softmax function normalizes the dense layer output in order to form a probability distribution over the alphabet $\mathscr{A}$. The dense layer comprises thus 20 units, corresponding to the size of the alphabet $\mathscr{A}$, including `GO` and `EOS` tokens. In total, the model contains 15,764 parameters to tune. The training is performed using `Keras` [24], by minimizing the categorical cross-entropy using the Adam optimizer (see [14, 24]), keeping 15% of the training set for validation. In generation mode, we sample letters, i.e. power cycle components, from the distribution given by the softmax activation function. All computations are carried out on Python 3.6.

The second part of the study is carried-out by finetuning the RNN model obtained previously. In particular, the architecture of the RNN model displayed in Figure 4 is kept unchanged, and so are the cost function, optimizer and training/validation ratio.

2. Learning the structure of a power cycle

Data augmentation The training process is initialized with a set $\mathscr{D}_0$ made of 450 distinct valid words randomly generated. In this case study, the words in $\mathscr{D}_0$ all contain at most one heat exchanger as well as at most one pair of splitter/mixer. 10 iterations of the procedure described in the previous section yield 53,959 unique words that can be converted into valid power cycles, with learning curve depicted in Figure 5. Thus, even if the initial dataset is very small, the iterative procedure succeeds in growing a dataset to a more reasonable size, along with an accurate model.

Although the alphabet $\mathscr{A}$ allows for up to 5 heat exchangers and 5 pairs of splitter/mixer, the generated words are all according to the initial dataset $\mathscr{D}_0$, i.e. with at most one of each of these two components. This illustrates a limitation of our data augmentation method: RNN are not able to go beyond the initial dataset, regardless of the alphabet, and cannot suggest for instance to use more heat exchangers. Hence, the shape of $\mathscr{D}_0$ strongly determines the

DOI: 10.17185/duepublico/48885

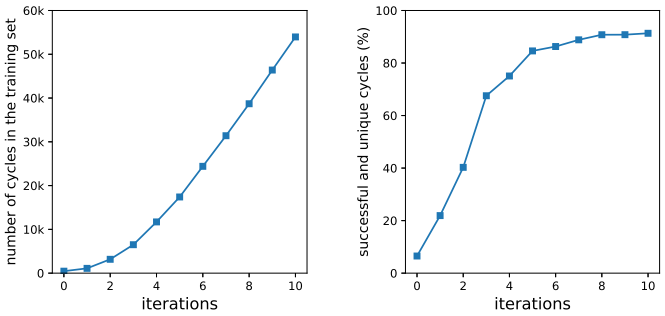

Figure 5: (Left) Number of power cycles in the training sets $\mathscr{D}_k$ and (right) proportion of valid and unique cycles generated by the RNN trained on $\mathscr{D}_k$.

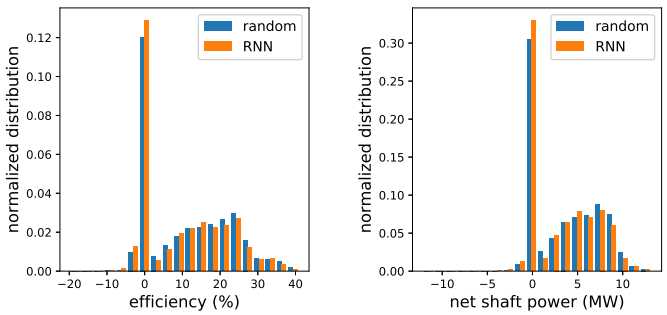

Figure 6: Distributions of the optimal efficiency (left) and net shaft power (right) achieved respectively for 11,744 cycles randomly generated (blue) and 9,116 cycles sampled by model $\mathscr{M}_1$ (orange).

kind of layouts produced afterwards by the RNN.

Language model Starting from the 53,959 distinct power cycles, of which 15% are kept for validation, we trained then another RNN model, denoted $\mathscr{M}_1$, with randomly initialized weights, and generated 100,000 words. Out of these 100k words, there are 93.1% valid words that can be converted into power cycles, out of which 87% are unique, of which 86% are new cycles. $\mathscr{M}_1$ is thus able to generate a large number of new layouts, and thereby to explore the space of SCO_2 cycles, within the boundaries set by $\mathscr{D}_0$ (e.g. no more than one heat exchanger). It has learned the language of power cycles, in the sense that 93.1% of the sequence of letters it generates do correspond to a power cycle layout from which a thermal efficiency can be computed. These statistics compare well with state-of-the-art RNN generative models trained on another problem, namely *de novo* drug design. See for instance the RNN in [25], which generates 94% valid words, out of which 90% are new. These results are all the more encouraging given that RNNs in drug design are trained on datasets of typically 10^5 to 10^6 samples, whereas our training set scales in 10^4.

In order to assess whether the power cycles generated by $\mathscr{M}_1$ are functionally similar to the original, random, database, we estimated the optimal shaft power $\mathscr{P}$ and thermal efficiency η for approximately 10,000 cycles in each one of the two collections, where the optimal design is obtained by maximizing the product $\mathscr{P} \times \eta$. This yields the normalized distributions drawn in Figure 6. The rather close agreement between the two distributions proves than $\mathscr{M}_1$ has successfully learned how to produce large sets of new power cycles, at the same time similar (in performances) and distinct (in layouts) from the original samples.

Figure 6 also underlines the need for finetuning, as it can be observed that a large share (43%) of the cycles generated by $\mathscr{M}_1$ actually consume more power than they are able to produce, and only a small proportion (less than 4%) have a reasonably good efficiency (greater than 30%). Again, RNN is intrinsically limited to its training set, and cannot improve it. However, this illustrates by contrast a key benefit of RNNs compared to random generators: while a random generator is bound to produce cycles all sampled from the distribution in Figure 6, RNN parameters can be subsequently modified to shift the distribution towards regions of higher values of η and $\mathscr{P}$. Hence the need for finetuning, to efficiently guide the exploration of the search space, addressed next.

3. Finetuned model

Targeting the design of a 10 MW CSP plant, we applied finetuning to produce cycles with $\mathscr{P} \times \eta > 300$, i.e. we aimed to generate cycles at least equivalent to a theoretical cycle with 30% efficiency and 10 MW net shaft power output (all the heat available from molten salt is absorbed by the theoretical cycle).

$\mathscr{M}_1$ is finetuned into $\mathscr{M}_2$ on a dataset containing 457 samples (15% kept for validation) with fitness $\mathscr{P} \times \eta > 300$. Because it is a small scale database, we adopt a training procedure similar to the first step. At each iteration, the model is finetuned for 5 epochs, with 5,000 cycles generated at each epoch. Nevertheless, the computational cost of evaluating the fitness of these 25,000 cycles, i.e. of solving a nonlinear optimization problem, is too heavy. To circumvent this issue, we trained a classifier on a separate database, prior to the finetuning. The task of the classifier is to assign every cycle in either one of two classes, namely $\mathscr{P} \times \eta > 300$ (positive class) or $\mathscr{P} \times \eta \leq 300$ (negative class). The prediction is formed based on the cycle layout only, and is not helped by any thermodynamic computation. At each iteration, the classifier replaces thus the fitness evaluation step, avoiding to solve the optimization problem. The outcome of the iterations is a pool of cycles predicted to be positive by the classifier. Using the classifer, fitness evaluation is replaced by fitness prediction. Actual fitness evaluation occurs then only when the iterations stop, in order to provide the actual designs of these good cycles.

Here, the classifier combines a LSTM layer with 32 hidden units and a 1-unit dense output layer with sigmoid activation. The training set contains 11,359 samples, of which 44% are positive. The performances of the classifier on a 1,000-sample test set are: 98% in accuracy (total error rate), 97% recall (fraction of true positive cycles that the classifier also predicts to be positive) and 98% in precision (fraction of predicted positive cycles that actually are positive). Although an important contribution of this work, the

DOI: 10.17185/duepublico/48885

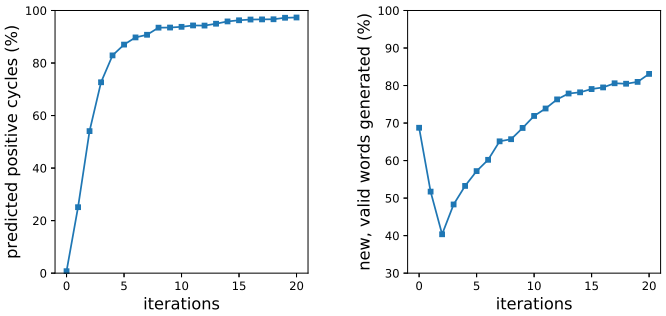

Figure 7: (Left) Fraction of valid generated cycles that are predicted to be positive ($\mathscr{P} \times \eta > 300$) and (right) fraction of the generated cycles that are both new and valid, during the course of the training of the finetuned model $\mathscr{M}_2$.

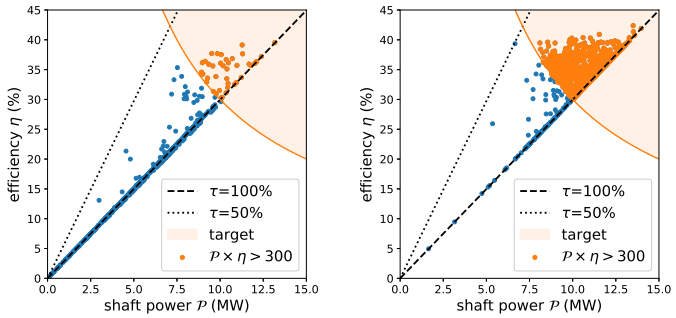

Figure 8: Performances $(\mathscr{P}, \eta)$ of 1000 cycles sampled from $\mathscr{M}_1$ (left, no finetuning) or $\mathscr{M}_2$ (right, finetuning). Cycles on the main diagonal have full salt use rate $\tau = 100\%$ and maximum amount of heat transferred from molten salt to the cycle ($\eta = \mathscr{P}/Q_{abs}$).

solution of this complex problem is not further described, for sake of brevity.

The training procedure results in 25,000 cycles per iteration, out of which those that have not been seen during previous iterations and that are predicted to be positive by the classifier form then the new training set for the next iteration of finetuning. The learning curve showing the performances of the finetuned RNN $\mathscr{M}_2$ at each iteration is displayed in Figure 7 (left). Similarly to the results in Figure 5, $\mathscr{M}_2$ is quickly able to generate a large proportion of cycles which will be detected as positive by the classifier - hence most likely with $P \times \eta > 300$. After a few iterations where the model loses some information acquired before finetuning, the final model generates almost only valid words not included in the training set, as shown in Figure 7 (right).

In total, 307,165 cycles have been predicted to be positive and valid during the finetuning process of the RNN. We proceeded then to the actual optimization of the fitness function $\mathscr{P} \times \eta$ of 17,212 cycles randomly selected in this set, in order to study their thermal performances. Figure 8 confirms the effectiveness of the finetuning: compared to the map obtained without finetuning (model $\mathscr{M}_1$, Figure 8, left), the cycles generated by $\mathscr{M}_2$ are almost all located towards region of higher values of $\mathscr{P} \times \eta$, beyond the boundary $\mathscr{P} \times \eta = 300$ (Figure 8, right). 86% of the 17,212 cycles actually are above the threshold $\mathscr{P} \times \eta > 300$, which is less than the precision of the classifier (98%). This difference might partially be explained by the fact that the optimizer only converges towards a local minimum, with no guarantee of reaching the absolute best value. Besides, predicting the fitness based on the layout information is a complex task. In particular, cycles on either sides of the boundary $\mathscr{P} \times \eta = 300$ might not be significantly structurally different from each other - observe that the negative cycles are mostly located close to the border (94% of the cycles are such that $\mathscr{P} \times \eta > 250$). On the other hand, without finetuning, less than 4% of the cycles are above the threshold. These results demonstrate how a finetuned RNN model can efficiently explore the search space and generate thousands of "good" power cycles.

By choosing the product $\mathscr{P} \times \eta$ as fitness function, we want to promote the design of cycles with both high efficiencies and high net power input. Unsurprisingly, this results in a majority of cycles with high salt use rate, where the salt use rate, or thermal energy storage (TES), τ, is the ratio $\tau = (T_2^\star - T_1^\star)/(T_2 - T1)$, with T_1, T_2 the maximum temperature range allowed for the molten salt, as in Table 1, and $T_2^\star, T_1^\star$ the effective temperature range of the molten salt achieved by the optimal design. With $\tau = 100\%$, the maximum amount of heat Q_{abs}^{max} is transferred from molten salts to the power cycle. Since $\eta = \mathscr{P}/Q_{abs}$, this means that for a fixed target power output $\mathscr{P}$, any cycle must have $\eta \geq \mathscr{P}/Q_{abs}^{max}$ - hence the shape of Figure 8, with every point above the dashed line $\tau = 100\%$. 63% of the tested cycles are such that $\tau > 0.9$. For other layouts, however, the best design finds a trade-off characterized by $0.6 \leq \tau \leq 0.9$. Figure 9 sheds light on this trade-off. In general, cycles with higher TES τ will produce more power $\mathscr{P}$. On the contrary, the efficiency η tends to increase with lower values of τ, consistently with the definitions of η and τ: smaller values of τ mean less heat Q_{abs} provided to the cycle, hence larger values of η. Besides, this trade-off has economical consequences. Indeed, having high TES τ enables to reduce the size of the storage system. On the other hand, lower values of η come with bigger solar fields, for a given target power $\mathscr{P}$. Hence, the criterion $\mathscr{P} \times \eta$ gives

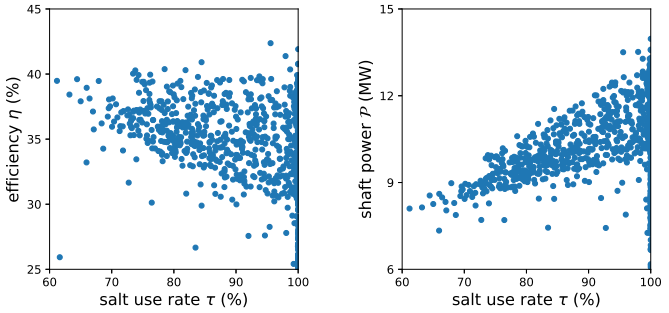

Figure 9: 1000 cycles in (left) the (τ, η) plane and (right) the $(\tau, \mathscr{P})$ plane, sampled from the RNN model after finetuning.

DOI: 10.17185/duepublico/48885

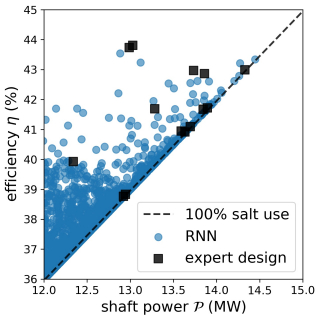

Figure 10: Best performances $\mathscr{P}, \eta$ obtained by the finetuned a RNN model $\mathscr{M}_2$ (blue circles) or by expert design (black squares)

a preview of what techno-economic design could result in. The best cycles in Figure 9 are those able to reach high TES while maintaining an elevated thermal efficiency, thereby decreasing the cost of both the storage system and the solar field.

Furthermore, observe that the generated cycles all have at most one heat exchanger, whereas it is well known that splitting the heat exchanger in two, respectively for the low and high temperature recuperations may help improving the thermal efficiency. Hence, most expert designs include one or two heat exchangers. Typical expert SCO_2 cycles, not described extensively here for sake of brevity are the Regenerative (RG, `TaACaH`), Recompression (RC, `Tab-1C1ACb1aH`), Pre-compression (PC, `TaCbACbaH`) and Partial cooling (PartC, `TabAC-1C1ACb1aH`). These fundamental cycles may be combined with standard improvement strategies, such as Intercooling (IC), Pre-heating (PH) and Reheating (RH). We compared the best designs achieved for these 16 state-of-the-art cycles (from RG-IC, RG,PH,... to PartC-RH), obtained by maximizing the same fitness function $\mathscr{P} \times \eta$, to those obtained with our method. It can be seen from Figure 10 that the best cycles sampled from the RNN model $\mathscr{M}_2$ perform equally well in the sense of the criterion $\mathscr{P} \times \eta$, and even slightly better for some of them, than the expert cycles. Although $\mathscr{M}_2$ was not trained with any *a priori* knowledge on the meaningful ordering of the components (the initial dataset $\mathscr{D}_0$ is random), it succeeds thus in challenging state-of-the-art expert designs.

From a more qualitative standpoint, the best layouts found in this case study are the words `TaACAC-1CH1ACa1H` and `TaAC-1CH1ACACaH1`, respectively with $\mathscr{P}, \eta, \tau = 43.3\,\%, 14.4\,\text{MW}, 100\,\%$ and $\mathscr{P}, \eta, \tau = 43.2\,\%, 14.3\,\text{MW}, 99\,\%$. Since these two words almost are the same, it means that both cycles are structurally very similar. See Figure 11 for a graph representation of `TaACAC-1CH1ACa1H` and `TaAC-1CH1ACACaH1`, confirming the closeness of these two cycles. In this experiment, the data-driven method slightly improves the expert design, in terms of best fitness function $\mathscr{P} \times \eta$. Yet, several findings are worth mentioning and show how relevant machine learning design is for cycle design. First of all, the best cycles displayed in Figure 11 actually both are a combination of PartC and IC, plus PH for `TaACAC-1CH1ACa1H`. This proves that the method

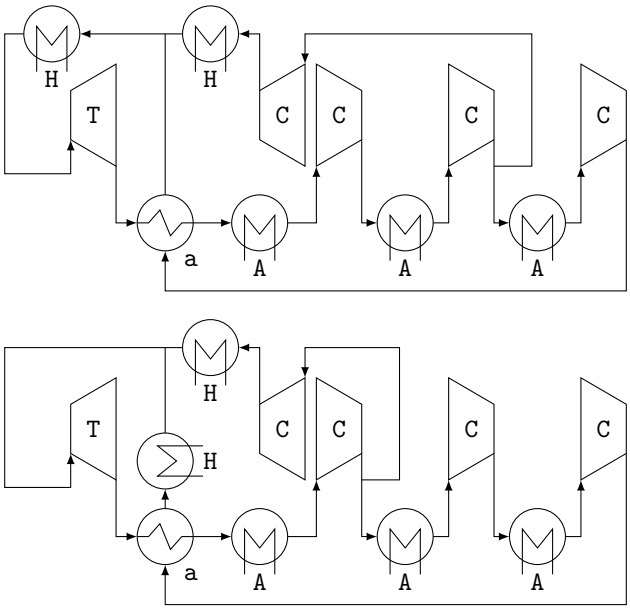

Figure 11: Layouts of the best cycles generated by the RNN, (above) `TaACAC-1CH1ACa1H` ($\eta = 43.3\%$, $\mathscr{P} = 14.4\,\text{MW}$) and (below) `TaAC-1CH1ACACaH1` ($\eta = 43.2\%$, $\mathscr{P} = 14.3\,\text{MW}$).

described in this work has gained physical insights despite being conceived without domain knowledge. The concepts of partial cooling, intercooling and preheating stand out, which is a valuable information for designing cycles with other constraints. On the other hand, $\mathscr{M}_2$ adds some creativity, observed for instance in the original heating scheme of `TaAC-1CH1ACACaH1` (see Figure 11, below). Besides, the main advantage of machine learning based design is that it can provide a whole population of (virtually infinitely many) *very good* cycles. Hence, the designer is able to pick a cycle that corresponds more closely to the constraints of a given project while maintaining equivalent level of performances. This profusion might be even more interesting when considering the techno-economic design.

Although computation time is not necessarily critical when dealing with power cycle design, our method suffers from a heavy computational cost. Because it is data intensive, the difficulty lies in the evaluation of the fitness function of a huge number of different power cycles. This limitation was also identified in other data-driven approaches, e.g. [11]. The numerical challenge varies with the layout of the cycle, with layouts containing at most one heat exchanger being more straightforward to simulate. The median of the 17,212 cycles sampled from the finetuned RNN is typically around one minute on one core of a 2.70 GHz, 16 GB-RAM laptop. Finding ways to efficiently solve the non-linear maximization of $\mathscr{P} \times \eta$ could thus increase the pace of exploration of the solution space.

CONCLUSION

This study aimed at revisiting power cycle design, with application to supercritical CO_2 CSP cycles, by adopting a data-driven

DOI: 10.17185/duepublico/48885

approach. Starting from a small set of random power cycle layouts, we used recent deep-learning techniques to train statistical models able

1. to imitate the initial database by generating new power cycles with similar structure and thermal properties (model $\mathscr{M}_1$);

2. to improve the initial database by producing specialized power cycles, all with a high thermal-to-electric efficiency and high net shaft power (model $\mathscr{M}_2$).

The main achievement is thus that the finetuned model $\mathscr{M}_2$ can produce large sets of distinct power cycles sharing good thermal properties. This could prove valuable to find a layout that suits best the peculiar constraints of a given project. Expert knowledge on power cycle design was not used at any step in the training of the deep learning model. In particular, the initial dataset $\mathscr{D}_0$ was randomly created, with no help of any domain expert. Yet, the successful cycles found in our experiments exhibit well-known patterns. For instance, partial cooling, intercooling and preheating schemes are included in the best layouts. The definition of the initial database is key to the process, because the model simply learns to imitate the cycles that belong to the training set. For the sake of demonstration, we chose here the most unfavourable setting by starting from a database with a limited number of power cycles, created at random. However, the constraint of randomness can be relaxed, by including expert layouts in $\mathscr{D}_0$. Beyond the few cycles discussed here (RG, RC, etc.), we can exploit already proposed superstructures to list many alternatives. Using the string representation of power cycles introduced in this article, it is easy to see for instance that the superstructure in [9] with 1152 configurations actually contains 450 unique cycles that can be added to $\mathscr{D}_0$.

In any case, it is to be noted that the definition of $\mathscr{D}_0$ is key to the process, because it will shape the distribution of the statistical model. In particular, the ratio of expert to random layouts in $\mathscr{D}_0$ controls the amount of expert knowledge transmitted directly to the model. If much greater than 1, the RNN language model $\mathscr{M}_1$ will most likely learn to infer the best concepts from those provided by the expert. On the contrary, if it tends to 0, as in this work, $\mathscr{M}_1$ will promote the best concepts from scratch - at the expense of more "waste" at the beginning of the training.

This preliminary study on machine learning based design of power cycle may be extended in several ways. One line of work consists in improving the fitness evaluation step, in terms both of convergence of the optimizer and efficiency of the solver used to perform the static simulation of the cycles. This would alleviate the computational cost of this expensive step, thereby opening the way to allow for more complex cycles to be included in the original database, e.g. with more than one heat exchanger.

Another promising extension is to proceed to the techno-economic design of SCO_2 cycles for industrial scale applications. This difficult task could indeed benefit from the large pool of thermodynamically similar but structurally different cycles generated by the RNN to lower the energy price of CSP.

Finally, let us mention that this work has enabled the creation of large datasets. They could be exploited to test alternative generative statistical models, such as the variational autoencoder [26], as well as more sophisticated finetuning strategies, for instance within reinforcement learning framework [25].

NOMENCLATURE

Acronyms	
CO_2	carbon dioxide
CSP	concentrating solar power
IC	intercooling
KNO_3	potassium nitrate
LSTM	long-short term memory
MS	molten salt
$NaNO_3$	sodium nitrate
PSO	particle swarm optimizer
PartC	partial cooling
PC	pre-compression
RC	recompression
RG	regenerative
RH	reheating
RNN	recurrent neural network
SCO_2	supercritical carbon dioxide
TES	thermal energy storage
Symbols	
`-1,-2,...`	tokens representing splitters
`1,2,...`	tokens representing mixers
$\mathscr{A}$	alphabet of power cycle components
`A`	token representing a cooler
`C`	token representing a heat exchanger
$\mathscr{D}_0$	collection of randomly generated cycles
$\mathscr{D}_k$	collection of cycles sampled from the RNN
`H`	token representing a heater
$\mathscr{M}_1$	RNN trained on a large dataset (language model)
$\mathscr{M}_2$	finetuned RNN model (specialized model)
$\mathscr{P}$	net shaft power output (MW)
P_c	compressor work (MW)
P_t	turbine work (MW)
Q_{abs}	heat duty (MW)
`T`	token representing a turbine
T_1	minimum allowed MS temperature (°C)
$T_1^\star$	actual minimum MS temperature (°C)
T_2	maximum allowed MS temperature (°C)
$T_2^\star$	actual maximum MS temperature (°C)
U,V,W	parameter matrices of a RNN
`a,b,...`	tokens representing a heat exchanger
h	hidden state of a RNN
o	output vector of a RNN
x	input vector of a RNN
Greek symbols	
δ	threshold for the fitness function
η	thermal-to-electric efficiency (%)
σ	activation function in a neural network
τ	salt use rate

DOI: 10.17185/duepublico/48885

REFERENCES

[1] V. Dostal, P. Hejzlar, and M. J. Driscoll, "High-performance supercritical carbon dioxide cycle for next-generation nuclear reactors," *Nuclear Technology*, vol. 154, no. 3, pp. 265–282, 2006.

[2] Y. Le Moullec, "Conceptual study of a high efficiency coal-fired power plant with co2 capture using a supercritical co2 brayton cycle," *Energy*, vol. 49, pp. 32–46, 2013.

[3] M. Mecheri and Y. Le Moullec, "Supercritical co2 brayton cycles for coal-fired power plants," *Energy*, vol. 103, pp. 758–771, 2016.

[4] T. Neises and C. Turchi, "A comparison of supercritical carbon dioxide power cycle configurations with an emphasis on csp applications," *Energy Procedia*, vol. 49, pp. 1187–1196, 2014.

[5] F. A. Al-Sulaimain and M. Atif, "Performance comparison of different supercritical carbon dioxide brayton cycles integrated with a solar power tower," *Energy*, vol. 82, pp. 61–71, 2015.

[6] J. Wang, Z. Sun, Y. Dai, and S. Ma, "Parametric optimization design for supercritical co2 power cycle using genetic algorithm and artificial neural network," *Applied Energy*, vol. 87, no. 4, pp. 1317–1324, 2010.

[7] V. T. Cheang, R. A. Hedderwick, and C. McGregor, "Benchmarking supercritical carbon dioxide cycles against steam Rankine cycles for concentrated solar power," *Solar Energy*, vol. 113, pp. 199–211, 2015.

[8] Y. Ahn, S. J. Bae, M. Kim, S. K. Cho, S. Baik, J. I. Lee, and J. E. Cha, "Review of supercritical co2 power cycle technology and current status of research and development," *Nuclear Engineering and Technology*, vol. 47, no. 6, pp. 647–661, 2015.

[9] Q. Zhao, "Conception and optimization of supercritical co2 brayton cycles for coal-fired power plant application," Ph.D. dissertation, Université de Lorraine, 2018.

[10] L. Wang, Y. Yang, C. Dong, T. Morosuk, and G. Tsatsaronis, "Parametric optimization of supercritical coal-fired power plants by minlp and differential evolution," *Energy Conversion and Management*, vol. 85, pp. 828–838, 2014.

[11] L. Wang, P. Voll, M. Lampe, Y. Yang, and A. Bardow, "Superstructure-free synthesis and optimization of thermal power plants," *Energy*, vol. 91, pp. 700–711, 2015.

[12] L. Wang, M. Lampe, P. Voll, Y. Yang, and A. Bardow, "Multi-objective superstructure-free synthesis and optimization of thermal power plants," *Energy*, vol. 116, pp. 1104–1116, 2016.

[13] T. Neveux, "*Ab-initio* process synthesis using evolutionary programming," *Chemichal Engineering Science*, 2018.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[15] "Smiles," http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html, accessed: 2018-11-23.

[16] M. Moran, H. Shapiro, D. Boettner, and M. Bailey, *Fundamentals of engineering thermodynamics*. John Wiley & Sons, 2010.

[17] I. H. Bell, J. Wronski, S. Quoilin, and V. Lemort, "Pure and pseudo-pure fluid thermophysical property evaluation and the open-source thermophysical property library coolprop," *Industrial & Engineering Chemistry Research*, vol. 53, no. 6, pp. 2498–2508, 2014.

[18] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, G. Varoquaux, T. Vaught, and J. Millman, Eds., 2008, pp. 11–15.

[19] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci. (MHS95)*. IEEE, 1995, pp. 39–43.

[20] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[22] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[25] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," *Journal of cheminformatics*, vol. 9, no. 1, p. 48, 2017.

[26] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules," *ACS central science*, vol. 4, no. 2, pp. 268–276, 2018.

DOI: 10.17185/duepublico/48885

DOI: 10.17185/duepublico/48885
URN: urn:nbn:de:hbz:464-20191002-174718-4