Diese Arbeit wurde vorgelegt am Lehr- und Forschungsgebiet
Theorie der hybriden Systeme

Generierung von probabilistischen Modellen für solarthermische Kraftwerke

Generation of probabilistic models for solar thermal power plants

Masterarbeit
Informatik

März 2016

Vorgelegt von Presented by	Levin Gerdes Henricistraße 44, 52072 Aachen Matrikelnummer: 297511 levin.gerdes@rwth-aachen.de
Erstprüfer First examiner	Prof. Dr. Erika Ábrahám Lehr- und Forschungsgebiet Theorie der hybriden Systeme RWTH Aachen University
Zweitprüfer Second examiner	Prof. Dr. Martin Frank Lehrstuhl für Mathematik (MathCCES) RWTH Aachen University
Korefferent Co-supervisor	Dipl.-Math. Dipl.-Inform. Pascal Richter Lehrstuhl für Mathematik (MathCCES) RWTH Aachen University

Eidesstattliche Versicherung

Gerdes, Levin
Name, Vorname

297511
Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit/Bachelorarbeit/~~ Masterarbeit* mit dem Titel

Generierung von probabilistischen Modellen für solarthermische Kraftwerke

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum

Unterschrift

Contents

Nomenclature

Symbols

D	Day of the year	
T	Time of day	[hrs]
t	Time of year	[hrs]
WD	Set of all weather data points	
SD	Set of all simulation data points	
K	Number of energy discretization classes	
k_{MRMG}	K for diffuse MRM gradient	
k_{TF}	Time frame size	
R	All state rewards are rounded to multiples of R	
p_{ij}	Probability of transition from state i to state j	
$\rho(s)$	Reward of state s	
S	State space of created model	
$\xi_p^{(1)}$	Weight for probabilities $p^{(1)}$ according to hourly expectancy	
$\xi_p^{(2)}$	Weight for probabilities $p^{(2)}$ according to weighted time frames	
$\xi_p^{(3)}$	Weight for probabilities $p^{(3)}$ according to diffuse MRM gradient	
Ξ_p	Weight vector for probabilities, $\left(\xi_p^{(1)}, \xi_p^{(2)}, \xi_p^{(3)}\right)$	
$\xi_r^{(1)}$	Weight for rewards according to hourly expectancy	
$\xi_r^{(2)}$	Weight for rewards according to weighted time frames	
$\xi_r^{(3)}$	Weight for rewards according to diffuse MRM gradient	
Ξ_r	Weight vector for rewards, $\left(\xi_r^{(1)}, \xi_r^{(2)}, \xi_r^{(3)}\right)$	

Acronyms

CSP	Concentrating Solar Power
DNI	Direct Normal Irradiation
DTMC	Discrete Time Markov Chain
MRM	Meteorological Radiation Model
MRMC	Markov Reward Model Checker
PV	Photovoltaic
SAURAN	Southern African Universities Radiometric Network
SUN	Stellenbosch University
SVN	Subversion
TMY	Typical Meteorological Year
UMIC	Ultra High-Speed Mobile Information and Communication
UVA	Ultraviolet A
UVB	Ultraviolet B
VM	Virtual machine

List of Figures

List of Tables

1 Introduction

The growing importance of renewable energy is not only a German phenomenon, it is apparent worldwide. In 2013, energy from renewable sources provided for 19.1% of the global final energy consumption and in 2014 comprised 58.5% of additions to the global power capacity. With the increasing electrification of transportation and heating in addition to advances in energy storage technologies, renewable energy provided enough energy to supply approximately 22.8% of the global electricity need [23].

According to estimations by the Renewable Energy Policy Network for the 21st Century (REN21), solar power accounted for 181.4 GW of the 657 GW renewable power capacity in 2014 (not including hydro). The solar power capacity consisted of 177 GW in solar photovoltaic capacity and 4.4 GW in concentrating solar thermal power.

With the aim of a reduction of CO2 emissions and since higher costs of fossil fuels are to be expected in the future due to the depletion and scarcity of natural resources, research in the fields of renewable energy in general and of concentrating solar power in particular intensified. The low investment cost per energy is one major positive aspect of concentrating solar power plants and renders them a target for closer considerations.

The PS10 (*spanish: Planta Solar 10*) in Spain, for example, is Europe's first commercial solar tower power plant, producing 11 MWe gross power of electricity and 23 GWh of electricity per year with an investment cost below 3500 € per kW [21].

During the planning process of such a solar tower power plant, one needs to assess the feasibility of locations in question. One major aspect is analyzing the climate and running test simulations. The requirement for weather data is often met using a so called Typical Meteorological Year (TMY), a set of mean weather data (including hourly values of solar radiation) yielding an artificial reference year for a specific location.

Since weather data is not always provided for more than just a few years, outliers and missing data have an inherently high impact on the creation of TMYs and the calculation of averages.

1.1 Task of this thesis

In this thesis, we propose a means of filtering the weather data and constructing a more flexible and reliable model for the simulation of solar tower power plants. We construct a model for the energy production of the Helio100 solar tower power plant in Stellenbosch, Western Cape Province in South Africa. For this site, the Southern African Universities Radiometric Network (SAURAN) provides measurements of weather data, beginning in the year 2010, on their website[1]. Since there only exists data of approximately six years, we cannot infer a reliable TMY. Instead, one might argue that a meteorological model alone offers a suitable solution. This approach does not consider given weather data, however, so we want to investigate whether we can

[1] `http://www.sauran.net/ShowStation.aspx?station=4`

construct a model based on given data.

Our model should incorporate the following key features:

- fitted to one specific solar thermal power plant,
- based on simulated energy production of real weather data,
- working well even for few data points, i.e. few years,
- readable by a Discrete Time Markov Chain (DTMC) model checker, and
- output energy values via checking for reward properties.

To this end, we need to prepare the provided measurements by means of identifying and handling (invalid) outliers. For identification purposes, we utilize the well established Meteorological Radiation Model (MRM).

The resulting set of weather data points is then used as the basis of the solar tower power plant simulation. We run a simulation of the Helio100 and all remaining weather data points using our own simulation model [10] and finally construct and analyze a probabilistic model based on the simulation values, which we may then use to calculate annual average amounts of energy received at the solar tower, as well as check for more involved properties using a model checker.

The layout of the heliostat field of Helio100 can be seen in Figure 1.

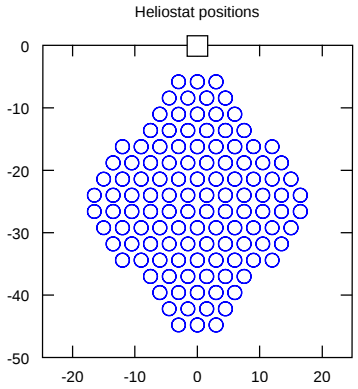


Figure 1: Solar tower field used for the simulation of the filtered weather data. The receiver is located at (0,0).

Later, we use the simulated energy production of this real weather data for verification of our model.

1.2 Related work

Of course, weather data is not only useful during the planning phase of power plants but is also valuable due to agricultural interests [25, 11], which renders accurate weather predictions an even more important field. Markov chains are already being used for

precipitation predictions [26, 24, 7]. Research concerning the optimum order ("memory length") of such models is undertaken [12] but usually focusses on stationary models, i.e., models without seasonal variation. In 1986, Amato et al. showed for four different time series of 20 years that solar irradiation sequences are in fact non-stationary [3].

To the best of our knowledge, there are no attempts at simulating either the solar radiation or the energy at the receiver of a solar tower power plant via the use of Markov chains based on real measurements.

Projects that come closest to this work are [24] and [2].Note though that in [24], Richardson proposes a stochastic simulation of the weather, including precipitation and radiation, but that only the precipitation prediction is done using Markov models. Conversely, in their work of 1988, Aguiar et al. utilize multiple Markovian transition matrices to simulation the global irradiation. However, they propose one solution for virtually any location on Earth which users can fit to their needs given they can produce the average monthly radiation for that location (or the average monthly number of sunshine hours), while we construct one or possibly multiple models for one specific location (and power plant configuration) and based on real solar radiation measurements.

1.3 Outline

In Section 2 we introduce the three main components which build the foundation to this work, (1) solar tower power plants, (2) computation of solar irradiation, and (3) probabilistic models. Section 3 describes the process of data preparation. This includes the sighting of all available weather data, reasoning behind the construction of a filter, and finally the resulting, filtered weather data. Section 4 comprises the main part of this work, explaining our approach to building the aforementioned probabilistic models as well as presenting first characteristics of the models themselves. The implementation part in Section 5 covers the most important aspects of the general program structure and the decisions that went into the design of our software solution. In the subsequent section, we evaluate our work and give a final conclusion in Section 7.

2 Preliminaries

In this section, we present preliminary concepts for our project. Section 2.1 explains the functionality of solar tower power plants and its main components. The following section presents the MRM we used for solar irradiation computation and ultimately for filtering a given set of weather data. Section 2.4 introduces core concepts of probabilistic models, DTMCs in particular.

2.1 Solar tower power plants

As opposed to Photovoltaic (PV) technology, the principle of solar tower power plants falls into the category of Concentrating Solar Power (CSP). The core concept of CSP is to localize the energy conversion in one single component instead of, e.g., converting directly inside each PV module. CSPs are further subdivided into line-focusing systems, such as linear fresnel collectors (Figure 2a) and parabolic troughs (Figure 2b), and point-focusing systems, such as solar dishes (Figure 3a) and solar tower power plants (Figure 3b). While line-focusing systems track the sun in one dimension only, point-focusing systems have to track the sun position in two dimensions.

(a) Linear Fresnel collector in 1.4 MW plant PE1 in Murcia, Spain [8].

(b) Parabolic-trough solar collector at Plataforma Solar de Almería, Spain [9].

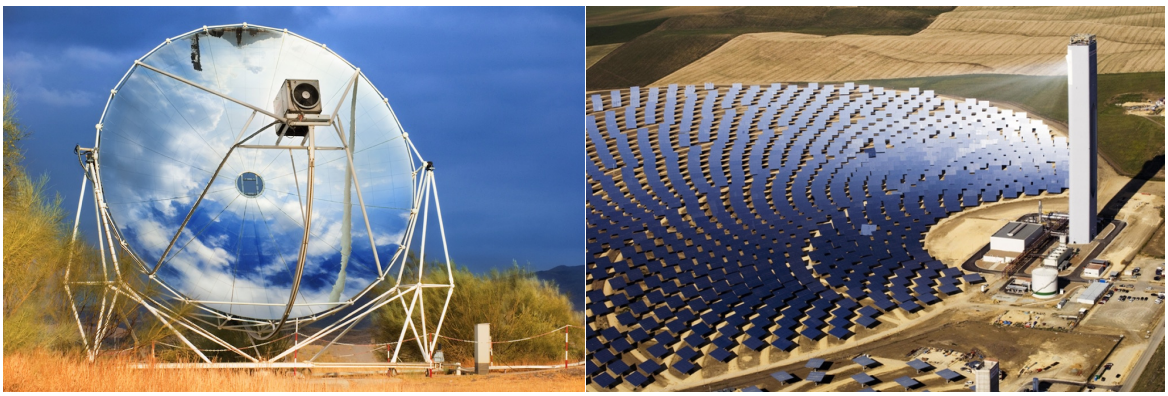
Figure 2: Line-focusing systems.

In the following we explain what a solar tower power plant consists of and how it produces energy.

It should be noted, however, that the approach presented in this thesis is applicable not only to solar tower power plants but also to the other CSP systems as well as offshore wind parks, given a simulation model which can output the energy production for a given time interval and weather condition.

2.1.1 Energy production

Solar tower power plants consist of a solar receiver, a heliostat field, a turbine, and a generator. Further components may differ depending on the distinct plant. Figure 4



(a) Dish-Sterling-System at Plataforma Solar de Almería, Spain.

(b) Solar tower power plant PS10, 11 MW in Andalusia, Spain.

Figure 3: Point-focusing systems. Sources: flickr, DLR.

depicts the general structure of a solar tower power plant.

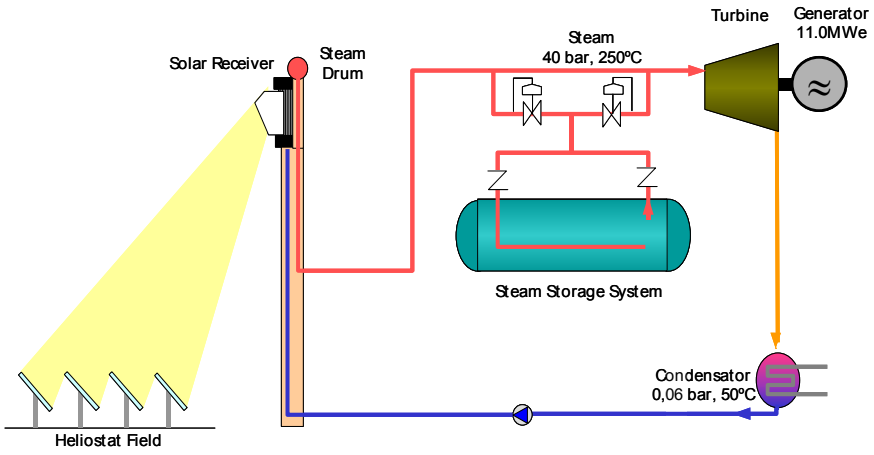


Figure 4: Concept of a solar tower power plant. This image is taken from [21].

The heliostat field consists of mounted mirrors which track the sun in two dimensions in order to reflect the direct incident sun beams onto the receiver near the top of the name giving solar tower. The direct sunlight is measured as radiant flux or power received by a surface per area in $^{W}/_{m^2}$, which is also known as irradiance. At the receiver, the concentrated sun beams heat up steam, air, or a fluid such as water, molten salt, or oil, which in turn exchanges heat with steam powering a turbine connected to a generator.

Finding an optimal layout for a solar thermal power plant is no trivial task as the heliostat field may vary greatly in terms of size, i.e., the number of mirrors in the hundreds or thousands, and the type of mirrors and focal lengths. Additionally the terrain may prove to be difficult and the heliostats can influence each other by shading

other heliostats or by blocking reflected sun rays depending on the sun's position, the heliostats, and the receiver location.

2.2 Simulation of a solar tower power plant

In this work, we simulate the solar thermal power plant Helio100 using the simulation model which is under development at the MathCCES [10]. The implemented ray-tracing methods follow a hierarchical approach [20, 6] which allows the detection of blocking and shading effects due to heliostats and the tower. The simulation model was verified using the Monte-Carlo tool SolTrace [27] and shows very good agreement.

2.3 Meteorological Radiation Model

In our work, particularly during the data preparation, we need to estimate solar radiation values. More precisely, we are interested in the solar irradiance per unit area which is incident on a plane perpendicular to the respective sun beam. This measure is called the Direct Normal Irradiation (DNI).

It is not a trivial task to assert the validity of weather station readings, since the DNI values vary not only during the day but also between seasons and years. Figure 5 shows simulated, hourly DNI values and solar positions throughout one year. One day is represented as an arc spanning from left to right representing the course of the sun during a day, i.e., it increases the solar azimuth and increases the solar altitude angle until noon and decreases this altitude angle from noon till the evening. The solar position at full hours of the corresponding day is marked and encodes the DNI via its color. A darker red signifies a higher DNI value. Multiple of these arcs with different widths comprise the entire year. A thinner arc has a lower maximum altitude and represents a day in the winter whereas a day represented by a wider arc corresponds to a day closer to or in summer. As can be seen in the graphic, the computed solar position changes, as expected, for the same hour of the day during one year (see the "eights" this creates in the plot).

Before entering the Earth's atmosphere, sun rays have a mean solar irradiance of 1366.1 $\mathrm{W/m^2}$ [4]. This value is called the *Solar Constant* and is not exactly a physical constant as it gets occasionally updated depending on ongoing measurements. While traversing the atmosphere, the DNI decreases due to different transmittances of the atmospheric gases. The global radiation gets divided into a diffuse and a direct part. With a changing solar position, the distance the sunlight travels through the atmosphere to a fixed point changes and therewith the DNI varies.

The MRM [18, 19, 15, 13, 22, 14, 4] provides a means to calculate the solar radiation under clear-sky conditions. Without interference by clouds we can thereby compute hourly high-valued estimates for the DNI which constitute the basis of the filter for the weather measurements later on.

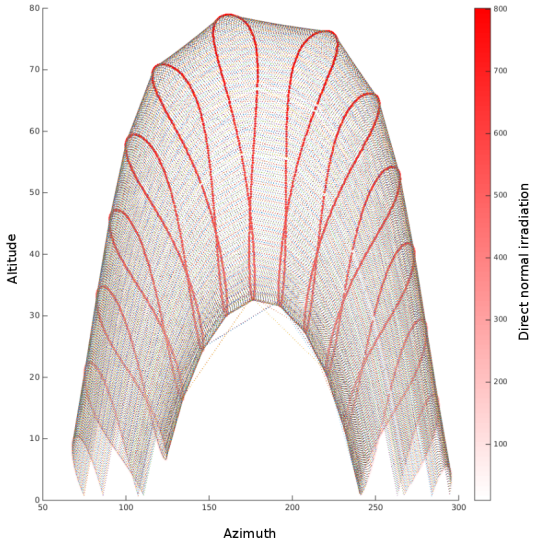


Figure 5: MRM data of one year. Hour points are highlighted.

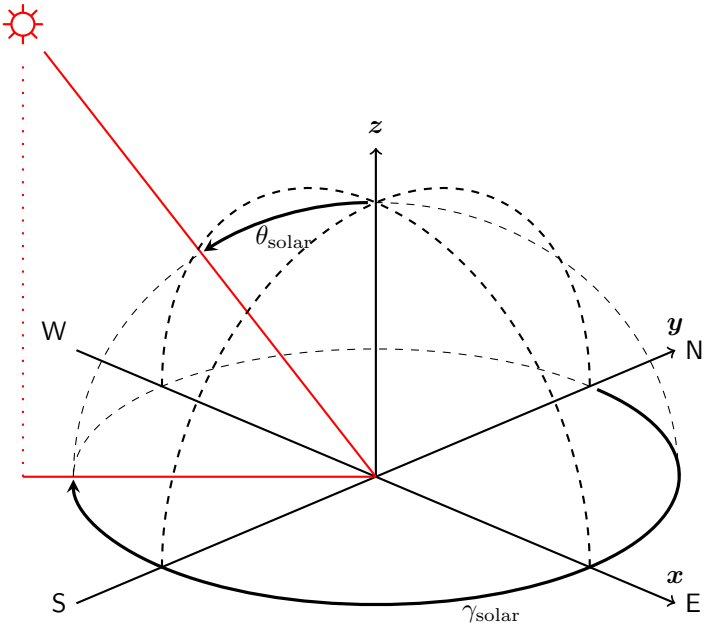


Figure 6: Solar angles.

2.3.1 Computation of the DNI

In the remainder of this subsection, we present the parts of the MRM which are necessary to compute the DNI as described in [4]. We deem this information useful as the mentioned reference on the one hand explains each aspect of the model, but on the

other hand does not explain the model as a whole for the latest version[2] of the MRM. Instead, it describes the initial model and then patches information for the updated versions. Additionally, we make notes whenever we find that the implementation, which was kindly provided by Dr. Kambezidis [14], deviates from the collated formulae.

The calculation of the DNI, I_{DNI}, depends on the extraterrestial irradiation I_{ex}, the solar elevation angle h and the optical transmittances T_i due to aerosol and Raeyleigh (molecular) scattering, ozone, water vapour, and mixed gases, respectively. Referring to Figure 6 and with θ_{solar} is the solar zenith in radian, the solar elevation angle is computed as $h = \pi/2 - \theta_{\text{solar}}$.

$$I_{\text{DNI}} = I_{ex} \cdot \sin(h) \cdot T_a \cdot T_r \cdot T_o \cdot T_w \cdot T_{mg} \tag{1}$$

Its unit is W/m^2. The extraterrestrial irradiance I_{ex} [W/m^2] is computed as

$$I_{ex} = I_{sc} \cdot \delta_{se} \tag{2}$$

where I_{sc} is the Solar Constant with a value of 1366.1 W/m^2 and δ_{se} approximates the distance between the sun and the earth depending on the day angle α_d:

$$\delta_{se} = I_{sc} \cdot \begin{bmatrix} 1.0 \\ 0.033 \end{bmatrix}^T \begin{bmatrix} 1 \\ \cos(\alpha_d) \end{bmatrix}. \tag{3}$$

Notice, that the literature computes δ_{se} more precisely as

$$\delta_{se} = I_{sc} \cdot \begin{bmatrix} 1.00011 \\ 0.034221 \\ 0.00128 \\ 0.000719 \\ 0.000077 \end{bmatrix}^T \begin{bmatrix} 1 \\ \cos(\alpha_d) \\ \sin(\alpha_d) \\ \cos(2\alpha_d) \\ \sin(2\alpha_d) \end{bmatrix}. \tag{4}$$

The day angle depends on the Julian day number d and is given in radian:

$$\alpha_d = 2\pi \cdot \frac{d-1}{365}. \tag{5}$$

Let m' denote the relative air mass,

$$m' = \frac{1}{\sin(h) + 0.50572(h \cdot (180/\pi) + 6.07995)^{-1.6364}}, \tag{6}$$

and P be the air pressure at station height and $P_0 = 1013.25$ hPa the air pressure at sea level. Then the absolute air mass is computed as

$$m = m' \cdot \frac{P}{P_0}. \tag{7}$$

[2]the latest MRM version as of March 2016 is MRMv5

Gas	A	B	C	D	l_i
H_2O	3.0140	119.300	0.6440	5.8140	l_w
O_3	0.2554	6107.260	0.2040	0.4710	l_o
CO_2	0.7210	377.890	0.5855	3.1709	350
CO	0.0062	243.670	0.4246	1.7222	0.075
N_2O	0.0326	107.413	0.5501	0.9093	0.28
CH_4	0.0192	166.095	0.4221	0.7186	1.6
O_2	0.0003	476.934	0.4892	0.1261	$2.095 \cdot 10^5$

Table 1: Coefficients for the optical air masses m_j.

For the gases listed in Table 1, the transmittances compute as

$$T_i = 1 - \frac{A \cdot m \cdot l_i}{(1 + B \cdot m \cdot l_i)^C + D \cdot m \cdot l_i}. \tag{8}$$

Using these, the transmittance due to mixed gases is computed as

$$T_{mg} = T_{CO_2} T_{CO} T_{N_2O} T_{CH_4} T_{O_2}. \tag{9}$$

In order to compute $T_w = T_{H_2O}$ and $T_o = T_{O_3}$, we need to find the values for l_w and l_o, respectively. While the literature states l_w as

$$l_w = 0.00493 \cdot e_s \cdot \frac{RH}{Temp}, \tag{10}$$

the code utilizes

$$l_w = 0.00493 \cdot e_s \cdot \frac{RH}{Temp} \cdot \left(\frac{P}{P_0}\right)^{0.75} \cdot \sqrt{\frac{273.15}{Temp}} \tag{11}$$

which depends on the saturation water vapour pressure [hPa] for the relative humidity RH [%] and the air temperature $Temp$ [K] at the station's height and computes as follows:

$$e_s = \exp\left(\begin{bmatrix} 22.329699 \\ -49.140396 \\ -10.921853 \\ -0.39015156 \end{bmatrix}^T \begin{bmatrix} 1 \\ (Temp/100)^{-1} \\ (Temp/100)^{-2} \\ (Temp/100) \end{bmatrix} \right). \tag{12}$$

Note though, that in the code, the last entry in the right-hand side vector has an exponent of -3 instead of 1. Furthermore, the code uses a different equation than (8), namely, it replaces all occurences of m with m' for the calculation of T_o and T_w.

The transmittance T_o concerning the ozone in the atmosphere is estimated using l_o which is calculated differently for each hemisphere if no measurements are available. Using the coefficients listed in Table 2, we can compute l_o in two steps:

$$l'_o = o_0 + o_2 \sin(o_3 \cdot (d + o_4)) + o_5 \sin(o_6(longitude + o_8)) \tag{13}$$

Hemisphere	o_0	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
Northern H.	150	1.28	40	0.9865	-30	20	$\frac{3\pi}{180}$	235	$\frac{20\pi}{180}$ if long. <0, 0 otherwise
Southern H.	100	1.5	30	0.9865	152.625	20	$\frac{2\pi}{180}$	235	0

Table 2: Coefficients for the calculation of l_o in Equations (13) and (14).

$$l_o = o_7 + l'_o \cdot \sin^2(o_1 \cdot latitude). \tag{14}$$

Regarding the initial problem (1), it only remains to compute the transmittance due to Rayleigh scattering,

$$T_r = \exp\left(-0.1128 \cdot m^{0.8346}\left(0.9341 - m^{0.9868} + 0.9391 \cdot m\right)\right), \tag{15}$$

and the transmittance due to aerosol scattering,

$$T_a = \exp\left(-m\beta\left(0.6777 + 0.1464m\beta - 0.00626 \cdot (m\beta)^2\right)^{-1.3}\right). \tag{16}$$

For the latter, we need the value of the Ångström's turbidity parameter, β, which is in the range of 0.05 to 0.4, where a lower β corresponds to a higher visibility range [km]. For instance, $\beta = 0.05$ corresponds to a visibility range of approximately 340 km, while $\beta = 0.4$ corresponds to a reduced visibility range of less than 5 km.

Again, if this value is not known, the MRM implements a means to estimate a β:

$$\beta = \beta' + \Delta\beta, \tag{17}$$

$$\beta' = 0.625 + 0.1\cos(latitude)\exp\left(\frac{-0.7 \cdot (station's\ altitude[\mathrm{m}])}{1000}\right), \tag{18}$$

$$\Delta\beta = \pm(0.02 \sim 0.06). \tag{19}$$

2.4 Probabilistic models

In model-checking, probabilistic models can be used to model systems which are subject to influences of stochastic nature, such as message loss or – in the context of this thesis – weather conditions.

A probabilistic automaton is a transition system which may define probability distributions for outgoing transitions from one state to all other states. Hence, the probability of taking a certain transition is given by the combination of the source and target state and the chosen probability distribution. In this section, we use the necessary definitions as introduced in [28].

Definition 1. *A sub-distribution over a countable set S is a function $\mu : S \to [0, 1]$ such that $\sum_{s\in S} \mu(s) \leq 1$; μ is a (probability) distribution if $\sum_{s\in S} \mu(s) = 1$. The set of all sub-distributions over S is denoted by SDistr(S), the set of probability distributions by Distr(S). By supp(μ) $= \{s \in S \mid \mu(s) > 0\}$ we denote the support of a (sub-)distribution μ.*

Different types of models can be distinguished by their time model (discrete or continuous) and by whether they exhibit non-determinism or not. Table 3 lists typical modeling approaches by this categorization. Efficient model checking algorithms exist for these probabilistic model classes and are implemented in (model checking) software such as PRISM[17], MRMC [16], and STORM[1]. The application field of such probabilistic model checking algorithms ranges from computer systems and security protocols to biological systems and quantum computing [28]. One reason is that the following problems cannot be modeled without a probability aspect [5]:

- randomized algorithms,
- modeling unreliable and unpredictable system behavior, and
- model-based performance evaluation.

Combining reachability analysis and model checking techniques with numerical mathematics, probabilistic model checking offers a means to automatically check for more intricate properties such as "the probability to reach a set of bad states is at most 1%" or "starting from a state s, the long-run average energy is at least 600 $^{\mathrm{Wh}}/_{\mathrm{a}}$" in addition to non-probabilistic model checking for qualitative correctness properties such as "is there any execution path which leads to a critical state".

Time	Nondeterminism	Probabilistic models
discrete	no	discrete-time Markov chains (DTMCs)
	yes	Markov decision processes (MPDs), probabilistic automata (PAs)
continuous	no	continuous-time Markov chains (CTMCs)
	yes	probabilistic times automata (PTAs), priced probabilistic timed automata (PPTAs)

Table 3: The types of probabilistic models currently supported by PRISM, classified by modeling of time and the presence of nondeterminism. Source: [17].

Regarding the differentiation between the model classifications, let Figure 7 be an excerpt of a transition system representing a probabilistic model. In this example, the state $s_{i,j}$ has three potential successor states, $s_{i+1,j+1}$ through $s_{i+1,j-1}$, where a transition with probability p_0 leads to $s_{i+1,j+1}$, another transition with probability p_1 leads to $s_{i+1,j}$ and so forth. Only if the probabilistic model behaves deterministically, the probabilities are known and sum up to 1. If the model features a discrete time model, a transition can only be taken after a fixed, discrete time interval, while a continuous time model generally allows transitions to be taken at any real time.

In this thesis we want to construct models with an hourly time discretization and based on measured weather data. In other words, we want to construct time discrete,

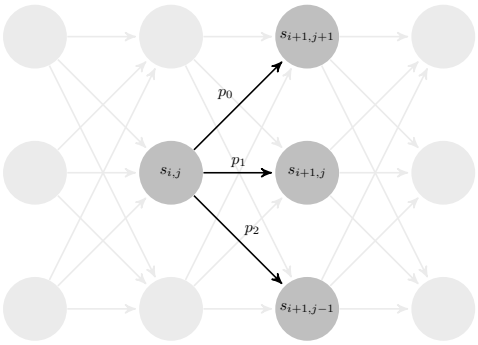


Figure 7: Excerpt of a transition system representing a probabilistic model.

stochastic (deterministic) models. Referring back at Table 3, we can see that a DTMC fits this description.

Definition 2. *A discrete-time Markov chain (DTMC) over atomic propositions AP is a tuple $\mathcal{D} = (S, s_{init}, P, L)$ with S being a countable set of states, $s_{init} \in S$ the initial state, $P : S \to SDistr(S)$ the transition probability function, and L a labeling function with $L : S \to 2^{AP}$.*

In order to fully utilize DTMCs in this thesis, we need to equip them with rewards representing the energy reception at the solar tower. In general, these rewards can be equipped to both the transitions and the states of the models. Using state rewards ($\rho : S \to \mathbb{R}$), we can then formulate (reward) properties which a model checker such as PRISM or STORM can check for.

One such property is the average or expected reward over all paths σ from a state s of at most length k through the model $\mathcal{D}$, i.e.,

$$ExpRew_s^{\leq k} = \sum_{\sigma \in \mathrm{Path}_s^{\leq k}(\mathcal{D})} P(\sigma) \cdot \rho(\sigma)$$

where $\rho(\sigma) = \sum_{i=0}^{k} \rho(s_i)$ is the reward and $P(\sigma) = \prod_{i=0}^{k-1} P(s_i)(s_{i+1})$ is the probability of a path $\sigma = s_0 \dots s_n$, $k \leq n$. We later use this property to compute the average annual energy received at the solar tower. Additionally, we write p_{ij} instead of $P(i)(j)$ for the probability of taking the transition from state i to state j.

Using these definitions, we do not have to concern ourselves with the memorylessness of the DTMCs and can focus on the construction of the same. For the evaluation of our models, we considered the three model checkers PRISM, MRMC, and STORM, which should compute the annual average energy as stated above. We discuss the reasoning behind choosing STORM in Section 5.5.1.

3 Data preparation

The first step after acquiring raw weather data is to identify and remove erroneous data, too high valued outliers in particular. Section 3.1 presents the original weather data and stresses its problematic aspects, thereby motivating the chosen means of filtering. The subsequent sections explain the concepts of the implemented filter, discuss the difficulties of finding fitting parameters, and finally show the resulting filter used in this thesis.

3.1 Available data

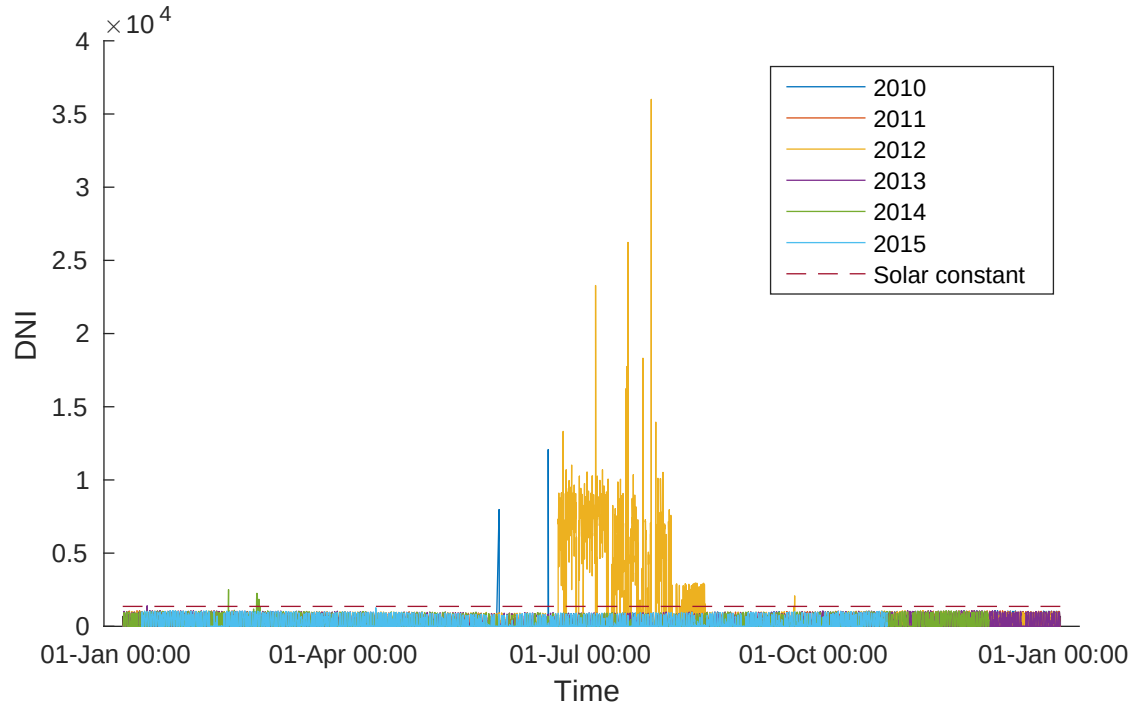
The SAURAN station at Stellenbosch University (SUN) (Figure 8) provides information about multiple solar irradiance measures: global horizontal irradiance, direct normal irradiance, and diffuse horizontal irradiance. Additionally, there are measurements of ultraviolet A (UVA), ultraviolet B (UVB), the air temperature, barometric pressure, relative humidity, and of the wind speed and direction.

Figure 8: Weather station on the roof of the SUN Engineering building. This image is taken from `http://www.sauran.net/ShowStation.aspx?station=4`.

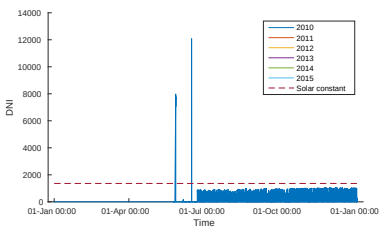
At the URL stated in Figure 8, minute-, hour-, and day-averaged data can be downloaded for a given time period. Although minute-averaged data is generally available from the day of installation on May 24, 2010 till the day of submitting this work in March, 2016, there is no complete data for any given year, i.e., there are empty-valued or missing minutes, days or even weeks. As stated in the station details document[3] of SAURAN, this can be due to sensor failures and maintenance work.

Figure 9 shows all available data from 2010 to 2015. Knowing that the theoretical maximum DNI value at the outer atmosphere is at 1366.1 W/m^2, multiple measurements are blatantly erroneous: the recordings during the (South African) winter of 2012 peak at 35994.2 W/m^2, more than 26 times the value of the Solar Constant. Except for 2011 and 2015, one can at first glance identify values exceeding this threshold for each of the data in the available years. Hence, we cannot simply ignore the entire year 2012 and use the remaining data, but we have to evaluate the feasibility of each measurement for the construction of the model individually.

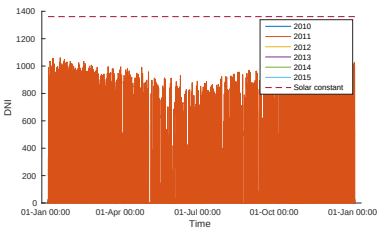
[3] `http://www.sauran.net/Docs/Sta_4/SUN%20Station%20Details.pdf`, last visited: 2016-02-06



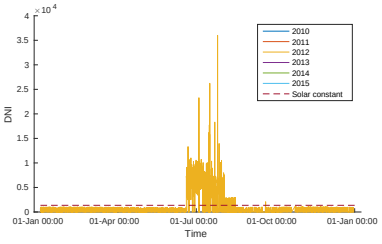
(a) All years



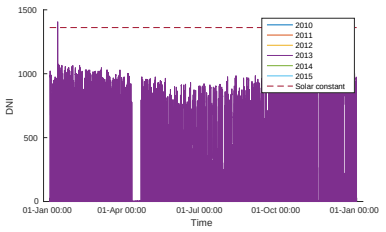
(b) 2010



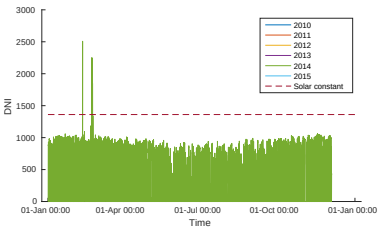
(c) 2011



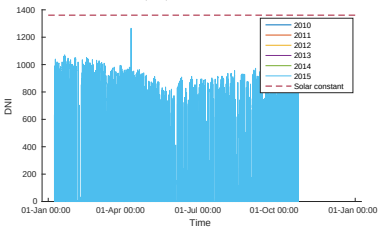
(d) 2012



(e) 2013



(f) 2014



(g) 2015

Figure 9: DNI values of all available weather data at the Helio100 site, depicted as DNI [W/m²] over time of the year.

It would be possible to simply trim all values at the Solar Constant. However, there are also false measurements with a magnitude within the mentioned boundaries but at the wrong time. As can be seen in Figure 10, there are measurements at night indicating a DNI at the level of late morning/evening.

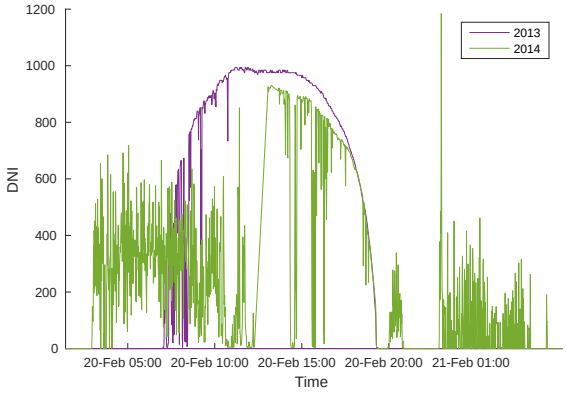


Figure 10: Erroneous nightly DNI measurements in 2014.

One way to cope with this problem is to simply calculate the sun angle and delete all positive records during the night and to cut off all values at 1366.1 W/m^2. But this solution would not solve all problems and it would even introduce new ones. A simple DNI cut-off at the Solar Constant produces fake data, while what we need to do is to delete the affected measurements. we have to delete the affected measurements instead. Having dealt with false night values and values exceeding the theoretical maximum extraterrestrial DNI, there may still be seasonal errors, e.g., measurements in the winter with too high DNI values for that time of the year.

In our endeavor to compute appropriate DNI bounds for a given time of the year, we utilize the MRM described in Section 2.3.

3.2 Filter

In order to perform plausibility checks on the measured DNI values, we compute upper boundaries for the given data points using the latest MRM according to the Clear Sky Model with a relative humidity of zero and low air pressure of 850 hPa as opposed to readings of up to approximately 1000 hPa, to achieve higher boundary values and thereby be less restrictive.

However, these settings alone do not suffice to compute feasible upper bounds. As can be seen in Figure 12a, the variance in daily DNI averages is too large (compare the winter months to the summer months). In order to reduce this variance, the origin location of the MRM can be moved closer to the equator, while it could not be moved farther, since it is apparent in all data sets that the winter months are around July.

We set the location of the MRM to 0 ° N latitude and 18.8654 ° E longitude, i.e., the same longitude as the radiometric station but on the equator (see Figure 11), to enable

the computation of greater values for the upper boundaries. Now each day has an upper boundary in the maximum of the corresponding MRM values (see Figure 12b).

Figure 11: Map of Africa depicting the location of the solar tower power plant Helio100 and the location assumed for the MRM for the computation of an upper bound. Graphic created using `http://www.darrinward.com/lat-long/`.

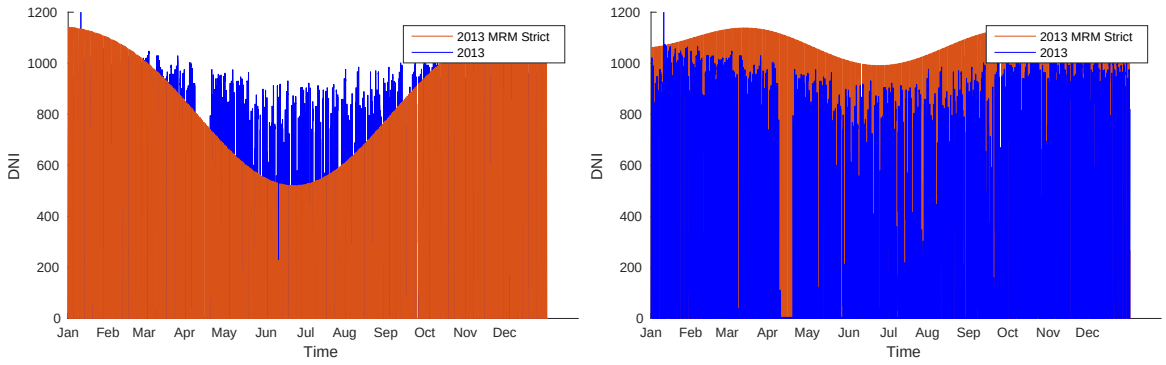
The MRM does not compute DNI values for each minute but only an average for each hour. To prevent our filter from discarding valid data points, we apply a relaxation to it: we allow the maximum MRM DNI of the previous and next k hours of the same day. We use DNI_x to denote the DNI value of data point x and $MDR(x, k)$ to denote the "MRM DNI Range" from point x and an integer k, sucht that, in other words, after applying our filter function f on the set of all weather data points WD, it holds that

$$x \in f(WD, k) \Leftrightarrow DNI_x \leq max\{DNI_{y \in MDR(x,k)}\} \wedge (\theta_{solar,x} < 90\,^\circ \vee DNI_x = 0)\,. \quad (20)$$

Let T_x be the time of day of x in hours with decimal minutes, starting at zero, T_y the hour of MRM data point y, also starting at zero, and D_x the day of year of x, starting at one, and D_y analogously the day of year of data point y. Now we can define MDR as follows:

$$y \in MDR(x, k) \Leftrightarrow \exists j \in \{-k, .., k\}.\lfloor T_x \rfloor + j = T_y \wedge D_x = D_y. \quad (21)$$

Figure 13 illustrates the difference between a strict filter and the relaxed filter.



(a) MRM origin at location of Helio100 (b) MRM origin on equator

Figure 12: Comparison of the influence different MRM origin locations have. Plot a shows the results of the site latitude and longitude, while b has the latitude set to zero.

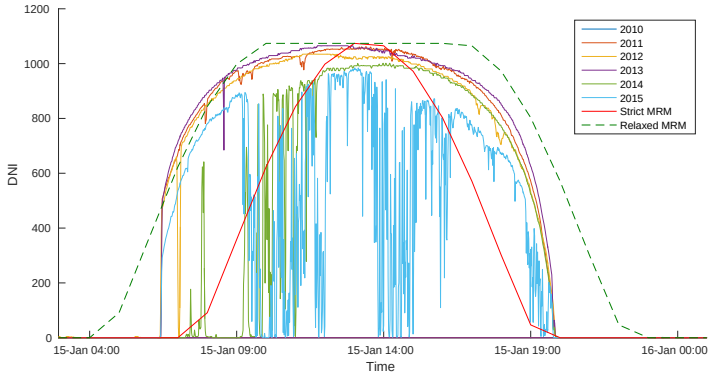


Figure 13: Relaxed and strict filter, without zenith oriented cut-off, illustrates the effect of MDR with $k = 3$. This graphic depicts the data belonging to a single day of the year.

The condition $\theta_{solar,x} < 90\,° \vee DNI_x = 0$ in Equation 20 reads as "*the sun's position must be above the horizon or the DNI must be zero*" and ensures that we delete nightly values even for large ks.

After the application of the filter, we expect the following positive effects:

1. There are no more outliers with extremely high values (some even surpassing the Solar Constant),
2. positive DNI values at night are discarded and
3. we allow higher values in the summer than in the winter.

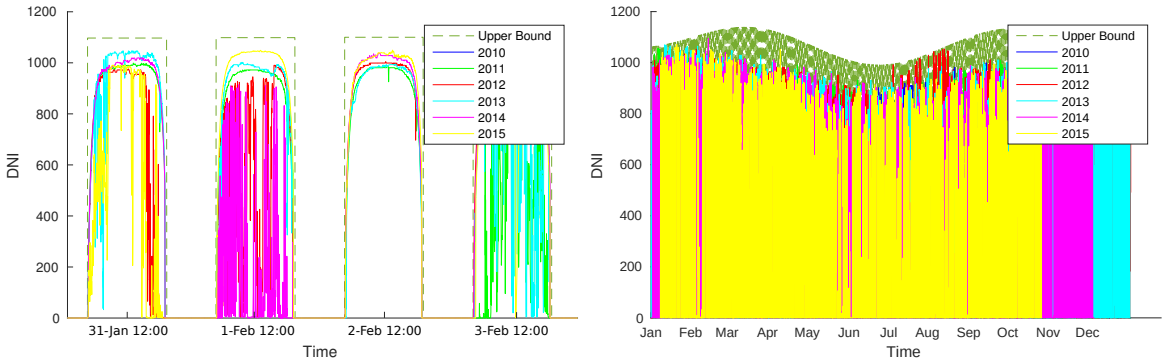
Note, that especially the latter cannot be achieved by simple thresholding with a constant.

3.3 Filter variants

As can be seen in Figure 13, the MRM filter needs to be relaxed by some hour-period k or it would delete high-valued data in the morning and evening, especially in the summer months. The maximum height of the filter is sufficient for each day, thus we advice against simple up-scaling of all boundary values.

Increasing the size of the MDR results in a plateau of the boundary at its daily maximum which approximates the appearance of the (near complete) energy curves of those days. Considering these plateaus and steep flanks, $k = 3$ seems to be a good choice. But in the evenings of summer days we still lose too many values we would like to keep (see Figure 13).

Increasing k's value further, and respecting the solar zenith, we get closer to a step function which is zero at night and has a single value greater than zero for the rest of each day. This daily value varies from day to day, and more importantly with the seasons, in such a way that it still allows the summer to maintain higher valued DNI entries than the winter days. Figure 14 illustrates these effects.



(a) Daily maximum boundary shown for four days.

(b) Varying daily boundaries, view on all days of the year.

Figure 14: Filter with $k = 12$.

Using $k = 12$, the filtered weather file contains 2651550 entries instead of 2557118 for $k = 3$. In other words, we can keep 94432 more minute-averaged DNI values, which make up 3.56% of all data points we deem correct. This shows that the filter using $k = 3$ really is too restrictive, which we already suspected in Figure 13.

The original weather data file contained 2767269 entries, partially complete with DNI values. Hence, the final filter with $k = 12$ gets rid of 115719 faulty data points, 4.18% of the input data.

4 Model construction

In the previous section, we have shown how our filter is constructed and that we can now assume a clean set of filtered DNI measurements. However, since our probabilistic model should not model the DNI [W/m^2] but the received energy [Wh] at the solar tower of the power plant Helio100, we have to compute this energy value for each minute based on the available (and valid) weather measurements. We achieve this by running the solar tower power plant simulation [10], which is under development at the institute MathCCES, for each weather data point remaining after the application of our filter. This results in a data set containing, among others, minute-wise energy values with the corresponding timestamps.

The DTMC models consist of states, rewards, and transitions. This section describes how each of these components is derived from the set of energy values.

The states and associated rewards are explained in Section 4.1 while Section 4.2 presents the construction of transitions and transition probabilities and therewith explains our models' approach to compensating for only few years' data being available.

4.1 States and rewards

To keep the models intuitive, their general layout is closely related to the available weather data. The index of a model state encodes approximations of the time of year as well as the received energy of a set of data points. This yields a two dimensional structure in which we represent a state as a tuple (t, e) for time t and energy e.

Note, that the weather data contains information about the DNI but that this dimension is not equivalent to the received energy which we want to encode in the model. Of course, the amount of received energy is not available for solar tower power plants (or other suitable generators) which are still in the planning phase. Hence the need for the solar tower simulation using the filtered weather data.

In order to compute the energy at the receiver of the Helio100 solar tower power plant, the simulation described in Section 2.2 is utilized. We simulate each minute of the filtered weather file individually.

Because there is no more than six years' data available for the application site, we will not represent each minute in the model, but discretize the time into hours and assign the data points according to their time of the year. This decision already decreases the influence of outliers because the energy encoded by a state[4] is now dependent on more data points as compared to having a state for each minute and energy class.

To further reduce the models' state space, the energy is discretized as well.

Here is an extreme example of how we can undermine the influence of outliers by the chosen time discretization: Assume that all data is valid and available for 59 minutes of an hour of the year for all six years, but in the remaining minute, only one data point is available, possibly an outlier with an energy value noticeably below the average of the remaining data of that hour but still within the same energy discretization class.

[4] or rather its reward

If we were to construct a state for each minute and energy class, the one outlier would dictate the (average) energy of that state. With a discretization to an hourly level, we use the average of $59 \cdot 6 + 1 = 355$ times the number of data points.

Now we can amend that (t, e) represents the state corresponding to time discretization class t (of 8784 hours, including leap years) and energy discretization class e (of K). This means, that $(t = 0, e = 1)$ is the state which represents the first hour (of every year) and the second lowest energy discretization class (see Figure 15).

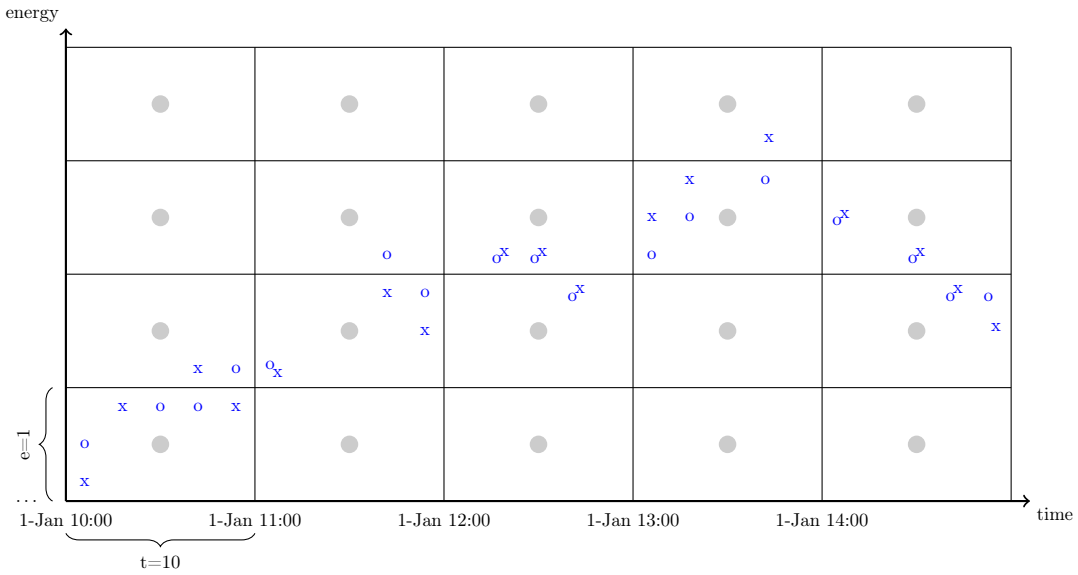


Figure 15: Discretization of the filtered data points (the graphic does not show real data points to keep the image clear and focus on the concept). Each data point is marked as an x. Every state (grey circles) represents a time and energy range. E.g., $(t = 0, e = 0)$ represents all data points in the first hour of each year and the lowest of K energy discretization class.

The general layout of the model can be seen in Figure 16. The time class index t increases from left to right, the energy class index e from the bottom up. The transitions will be explained in Section 4.2. For now, it suffices to know that there will be no transitions from right to left, no purely vertical transitions, no transition increasing t by more than one at a time, and no loops except for a self-loop in the final state. All paths through the model traverse the model states in a manner which increases t and may vary the value of e.

Let S denote the set of all states and $Suc(i \in S) \subset S$ the set of all potential direct successor states of i in paths through the model, i.e., $Suc(i)$ is the set of all states $j = (t', e')$ with $t' = t + 1$ for a state $i = (t, e)$.

A possible year is then represented as a path from the initial to the final state, during which the sum of state rewards on that path is received at the power plant's solar tower.

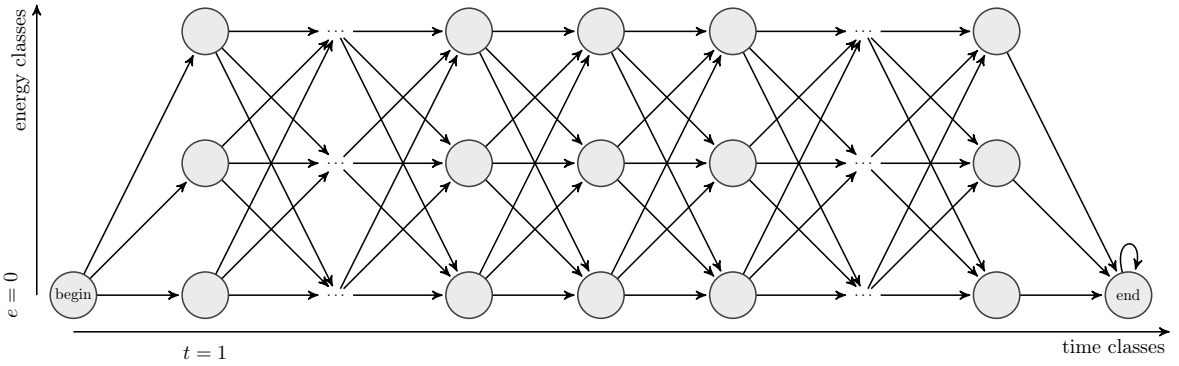


Figure 16: General layout of a model with three energy classes.

Rewards. It remains to decide how to define rewards for the models. As stated in the preliminaries, rewards can be associated with both the states and the transitions. We believe that the more intuitive way is to associate them with states since they are directly derived from the time and energy values and not from the tuples of states and the probabilities of reaching one another. For better readability, we simply write $reward(s)$ instead of $\rho(s)$ in this section.

A state's reward could simply be defined as the energy at the center of the corresponding energy discretization class e, i.e., the reward for a state could be set to

$$reward(s = (t, e)) = e \cdot \frac{\text{max energy } [\text{W}/\text{m}^2]}{K}. \tag{22}$$

However, with this approach all states sharing the same e are assigned the same reward. This is problematic when the minute-averaged energy values in state (t, e) are near the upper boundary of e while the energy values in another state (t', e) are closer to the bottom boundary. The reward would be the same even though it is apparent that the reward of (t, e) should be higher than that of (t', e). Of course, one could increase the number of energy discretization classes to reduce this discrepancy, but the underlying problem remains the same.

A more accurate solution is to choose, for each single state, the average energy of all data points belonging to that state. Let SD_s denote the set of all simulation data points associated with state s, i.e.,

$$x \in SD_{s=(t,e)} \Leftrightarrow \lfloor \text{time of year in } x \rfloor = t \wedge \frac{\text{received energy in } x}{\text{energy discretization step size}} = e. \tag{23}$$

This leads us to the following equation:

$$reward\,(s) = \frac{\sum_{x \in SD_s} \text{received energy in } x}{\mid SD_s \mid}. \tag{24}$$

But even that is not sufficient. As mentioned earlier, we simulate every minute separately and got energy values in the unit Wh/min. The time discretization however changed the time resolution to one hour, hence the need for a multiplication of the

state rewards with 60. Thus, instead of Equation (22) or Equation (24), we will use Equation (25):

$$reward\,(s) = \begin{cases} 60 \cdot \frac{\sum_{x \in SD_s} \text{received energy in } x}{|SD_s|} & \text{if } |SD_s| \neq 0 \\ 60 \cdot (e_s + 0.5) \cdot \text{energy step size} & \text{otherwise} \end{cases}. \tag{25}$$

4.2 Transitions

This section presents how the transitions, and more precisely the transition probabilities, are derived from both the measured and the simulated data.

We denote the probability of taking the transition from state i to state j as

$$P_{ij} = \xi_1 \cdot p_{ij}^{(1)} + \xi_2 \cdot p_{ij}^{(2)} + \xi_3 \cdot p_{ij}^{(3)} \tag{26}$$

with $\sum_{i=1}^{3} \xi_i = 1$.

We only allow transitions from left to right and only to a state j which represents an energy class e' in the hour t', which follows directly after hour t in state i. This means that for all transition probabilities, the following formula has to hold:

$$p_{ij} = \begin{cases} 0 \leq (p \in \mathbb{R}) \leq 1 & \text{if } \exists e, t, e', t' \text{ s.t. } i = (e, t) \wedge j = (e', t') \wedge t' = t + 1 \\ 0 & \text{otherwise} \end{cases}$$

with $\forall i \in S : \sum_{j \in Suc(i)} p_{ij} = 1$ and $\sum_{j \notin Suc(i)} p_{ij} = 0$.

4.2.1 Hourly expectancy

The first probability measure is derived from computing the average or expectancy of the given data. The probability of transitioning to a state representing many data points should intuitively be higher than that of a state representing fewer data points.

Hence, we define the hourly expectancy probability of the transition from state $i = (t, e)$ to $j = (t + 1, e')$ as

$$p_{ij}^{(1)} = \frac{\#\text{ data points at } (t+1, e')}{\#\text{ data points at time } (t+1)}. \tag{27}$$

Figure 17 illustrates this approach for a small number of data points and states. Note that the probabilities of all outgoing transitions from state (t, e) are independent of e.

If the number of data points at time $t+1$ is zero, we simply hold the current energy value under the assumption that the chance of multiple subsequent time columns being without any data points is very low. Hence, for this case we say $p_{(t,e),(t+1,e)}^{(1)} = 1$ and $p_{(t,e),(t+1,e')}^{(1)} = 0$ for $e' \neq e$.

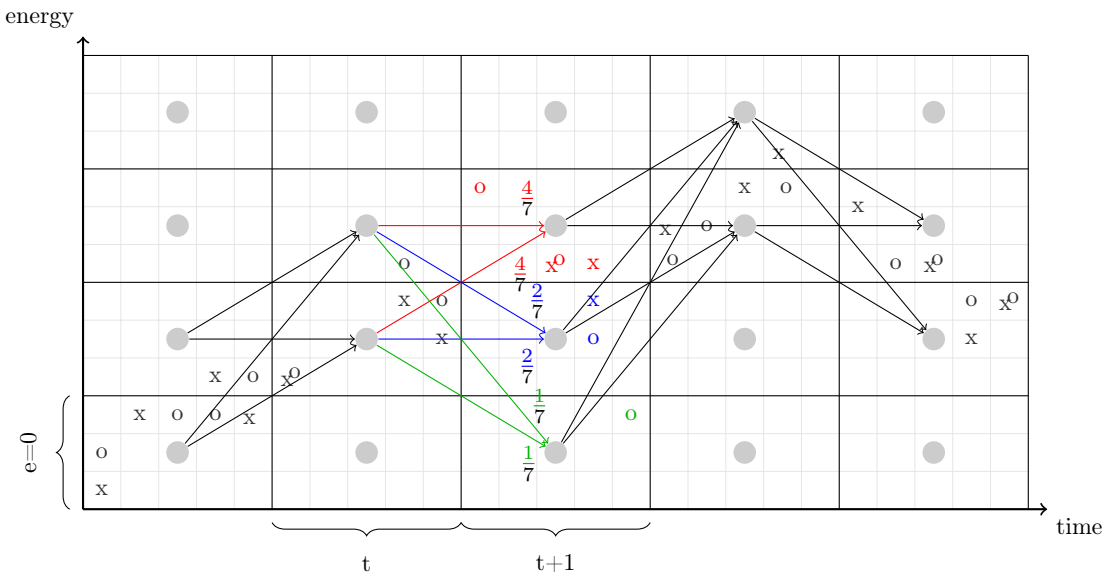


Figure 17: Conceptual probability computation according to hourly expectancies. x and o represent data points of two different years while the colors indicate which enumerator is influenced.

4.2.2 Time frame probability

There may be time frames in which only very few or even no data points can be found, which can occur due to, e.g., sensor failures or maintenance operations. In these cases the model cannot reliably model the respective time frame.

For example, if there is only a single data point for an entire hour, then the hourly expectancy-method as described above would set the probabilities of all in-going transitions of the corresponding state $s = (t, e)$ to 1 (and all transitions to $s' = (t, e')$ to 0 for all energy classes $e' \neq e$). The same effect may of course take place for entire days.

In order to reduce the impact of such outliers and to also provide sensible probabilities, we propose to also regard the *time frame data*.

The intuition behind this approach is that the energy curves of neighbouring days are similar so that we may to some extent compensate a possible lack of real data. This approach can be thought of as a smoothing operation.

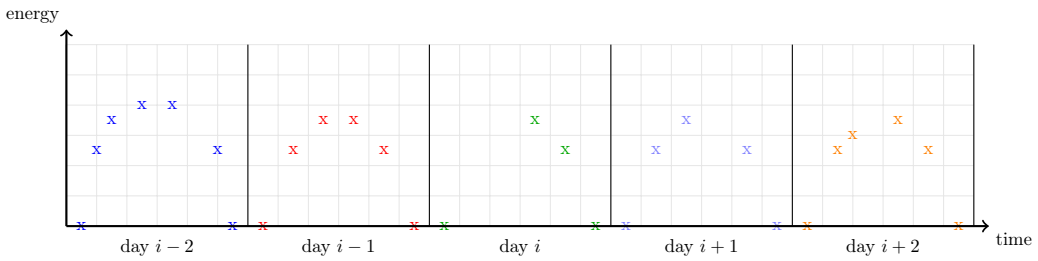
The time frame data is the set of all data points x in a time frame of size k, e.g., 7 days to either side of a day of the year D.

$$TF_k(D) = \{x \in SD \mid D - k \leq D_x \leq D + k\} \tag{28}$$

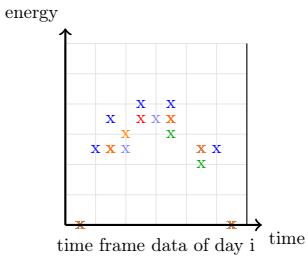
where, again, $D_{x \in SD}$ denotes the day of year for data point x.

With $D_t = \lfloor t/24 \rfloor$ being the day of the year to which t belongs, we can then define the time frame data of one hour t of the year as

$$TF_k(t) = \{x \in TF_k(D_t) \mid t_x = t\}. \tag{29}$$



(a) Original data containing few data points for the center day D_0.



(b) Time frame data for $k_{TF} = 2$.

Figure 18: Concept visualization of time frame data. The data points in (a) show five days' data points, two days to either side of a day D_0. The course of the daily energy gets smoothed as more data is available for average computations, which can be seen in (b).

Figure 18 illustrates the approach for a time frame of five days, i.e. $k = 2$. In order to better understand the *weighted time frames* later on in this section, it is important to see that we do not change any timestamps: we do not *move* any data points but instead only make note of their existence and time and energy values. They belong to (multiple) time frames but remain unchanged.

Let $TF_k(j \in S) \subseteq TF_k(t_j)$ be the set of all data points in the time frame around t_j which have an energy value in class e_j. Similar to $p_{ij}^{(1)}$, we can then calculate the time frame probability as

$$p_{ij}^{(2)} = \frac{|TF_k(j)|}{|TF_k(t_j)|}. \tag{30}$$

And again, as for the hourly expectancy, we hold the current energy value if there is no data point in the next time class, or, in other words, if $|TF_k(t_j)| = 0$.

Similar to Equation (25), the rewards for the time frame approach are computed as

$$reward\,(s) = \begin{cases} 60 \cdot \frac{\sum_{x \in TF_k(s)} \text{received energy in } x}{|TF_k(s)|} & \text{if } |TF_k(s)| \neq 0 \\ 60 \cdot (e_s + 0.5) \cdot \text{energy step size} & \text{otherwise} \end{cases}. \tag{31}$$

Note that the probabilities computed by this method do not have to be the final probabilities used in the models, but only have a weighted influence on it such that we can reduce or increase the smoothing effect as desired.

Weighted time frame. Since the differences in energy curves increase with a growing distance from the day D_0 in the center of the time frame, we want those days closer to the border to influence the time frame data less than the days in the center.

Our proposed solution is a weight function with a maximum in D_0. Data points closer to D_0 will be associated with greater weights and will thus have a greater influence on the probability $p_{ij}^{(2)}$. In this work, the weighted time frame data $TFW_{k,w}(D)$ is noted as a pair over the un-weighted time frame data $TF_k(D)$ and a weight function $w_{k,D}$:

$$TFW_k(D) = \langle TF_k(D), w_{k,D} \rangle . \tag{32}$$

We use k and k_{TF} interchangeably in this section to increase the readability in formulae.

The foundation of the weight function is a triangular (or tent) function

$$\text{tri}\left(\frac{x - x_0}{l}\right) := max\left\{1 - \left|\frac{x - x_0}{l}\right|, 0\right\} \tag{33}$$

with a base length of $2 \cdot l$ (later: twice the number of days) and the single global maximum at $(x_0, 1)$. Note that only in this equation x does not stand for a data point but for just an arbitrary element of the general triangular function's domain.

The height of the triangle at a given day of the year equals the weight of the data points of that day and can, in case of an integer weight, be thought of as the multiplicity of those data points. Since we want to include the days $D = D_0 \pm k$ and the weight at the base of the triangle is zero, l must be equal to $k + 1$. If we want the weights to represent data point multiplicities, they must be non-negative integers. We can achieve this by multiplying with the only denominator in the equation, $l = k + 1$.

$$(k+1) \cdot \text{tri}\left(\frac{D - D_0}{k+1}\right) = max\left\{k + 1 - |D - D_0|, 0\right\} \tag{34}$$

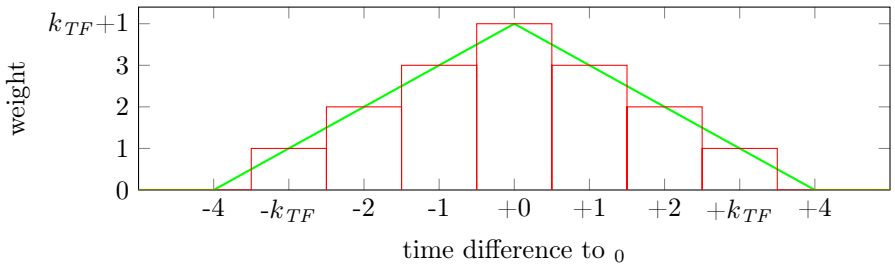


Figure 19: Time frame weight function for $k_{TF} = 3$.

Using the set of time frame data points $TF_k(D)$, the weight function $w_{k,D} : TF_k(D) \to \mathbb{R}_{\geq 0}$ can be defined as follows:

$$w_{k,D_0}\left(x \in TF_k(D_0)\right) = (k + 1 - |D_x - D_0|) \tag{35}$$

Figure 19 depicts the resulting weight function for $k_{TF} = 3$. Using such a weight function, we can calculate the influence a day's data points have on the probability calculation for the states within the day in the center of a time frame, D_0.

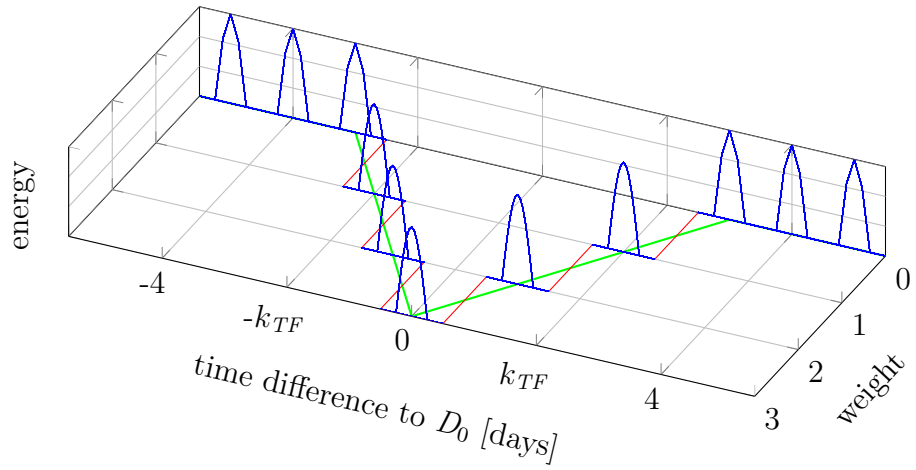


Figure 20: Weighted time frame concept for $k_{TF} = 2$.

Figure 20 illustrates these effects: the current D_0 has the highest weight $(k_{TF} + 1)$, the neighbouring days' data points have gradually less weight until neighbour days $D_0 \pm k_{TF}$ are reached, which have no influence at all. The energy values of the data points is untouched however.

It still remains to calculate probabilities using the weighted time frame data. Because we do not want to change energy values, as they are utilized for the reward computation, the probabilities will only rely on the weights and existence of data points. Using the weight function in Equation (35), one can imagine that we multiply a data point by its weight according to the current weighted time frame. With this image in mind, the following formulae are analogue to those for the hourly expectancy computation.

For easier notation later on, we define the weighted time frame data of one hour t to be a subset of $TFW_k(D_t)$ by listing $TF_k(t)$ instead of $TF_k(D_t)$ as the first element of the pair:

$$TFW_k(t) = \langle TF_k(t), w_{k,D_t} \rangle . \tag{36}$$

The weight of "column" t in the weighted time frame is then calculated as follows:

$$|TFW_k(t)| = |\langle TF_k(t), w_{k,D_t} \rangle| := \sum_{s' \in TF_k(t)} w_{k,D_t}(s'). \tag{37}$$

In order to get the time frame value for a single state $s = (t, e)$ instead of the time frame value for an entire time unit t, we specify a weight function based on s, and of course k_{TF}. This function uses the weight of Equation (35) if the energy class of s' is e, the same energy class as that of s.

$$w_{k,s \in S}(s' \in TF_k(D_t)) = \begin{cases} w_{k,D_t}(s') & \text{if } e_s = e_{s'} \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

Similar to Equation (37), the weight of a state $s = (t, e)$ is then calculated as

$$|TFW_k(s \in S)| = |\langle TF_k(s), w_{k,s} \rangle| = \sum_{s' \in TF_k(D_s)} w_{k,D_s}(s'). \tag{39}$$

Using this, the previous time frame probabilities in (30) may now be equipped with weights. A transition's probability is computed similarly to the hourly average probabilities, but takes all points of the weighted time frame into account. Hence, the probability of a transition from state i to j is the weighted number of data points in the time frame at time t_j and energy class e_j divided by the total weight of data points at time t_j:

$$p_{ij}^{(2)} = \frac{|TFW_k(j)|}{|TFW_k(t_j)|}. \tag{40}$$

The reward computation for the (weighted) time frame approach remains unchanged (Equation (31)).

4.2.3 MRM gradient

Because the data represents only a few years, there is a need for an orientation based on ideal weather and energy curves. Hence, the probability we represent in this section only takes the MRM into account. Although the problem does not model the DNI per hour but the produced energy per hour, one can see that the trends of both irradiation and energy output are related and alike.

The general idea is, that it is reasonable to observe an increase in energy values from dawn to midday, and a decrease toward dusk. The MRM is used to approximate artificial energy curves of each day.

The probabilities are derived from the MRM gradient: assume that from hour t to $t+1$ the MRM DNI value raises $40\,\mathrm{W/m^2}$. We then translate this increase to an increase in the index of energy discretization classes.

$$\nabla_{MRM}(t) = \frac{+40\,\mathrm{W/m^2}}{\text{irradiation step size}} = \frac{+40\,\mathrm{W/m^2}}{\frac{\text{max DNI}\,\mathrm{W/m^2}}{K}} = K \cdot \frac{+40}{\text{max DNI}} \tag{41}$$

According to this form of MRM gradient probabilities, every path through (t, e) should also contain $(t+1, e+\nabla_{MRM}(t))$ for every hour of the year t, $0 \leq t \leq 8784$.

Since we apply this to all energy classes, an imminent problem is, of course, that $e+\nabla_{MRM}(t)$ might be smaller than e_{min} or greater than e_{max}. For convenience, let $e' = e+\nabla_{MRM}(t)$. If $(t+1, e')$ does not exist because $e' > e_{max}$, then set $e' := e_{max}$. Analogously, set $e' := e_{min}$ if it were otherwise smaller than e_{min}.

Figure 21 visualizes this approach. The red lines correspond to the course of the MRM through the respective discretization classes while the black lines represent the model transitions with a probability of one. For example, in the first time step, the MRM gradient is +1. Thus all outgoing transitions of the states at the first hour are equal to one, if and only if the transition leads to a state in the next highest energy discretization class (if possible, note the highest state and its successor with the same energy discretization class).

More formally, $p_{ij}^{(3)}$ can be defined as follows:

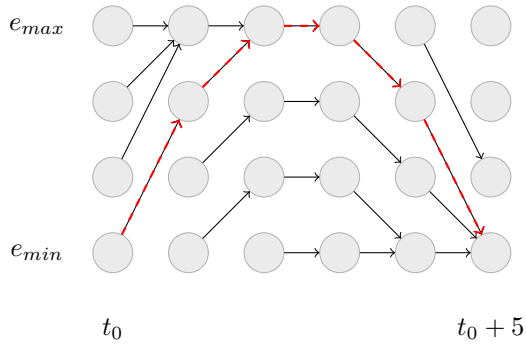


Figure 21: Concept of the basic MRM gradient probabilities.

$$p_{ij}^{(3)} = \begin{cases} 1 & \text{if } \exists t, e, e' : i = (t, e) \wedge j = (t+1, e') \wedge e + \nabla_{MRM}(t) = e' \\ & \quad \wedge e_{min} \leq e' \leq e_{max} \\ 1 & \text{if } \exists t, e : i = (t, e) \wedge j = (t+1, e_{max}) \wedge e + \nabla_{MRM}(t) > e_{max} \\ 1 & \text{if } \exists t, e : i = (t, e) \wedge j = (t+1, e_{min}) \wedge e + \nabla_{MRM}(t) < e_{min} \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

Contrary to utilizing only a combination of the hourly expectancy and *time frame probability*, the *MRM gradient* makes use of the underlying DTMC, since without the latter, the probabilities of outgoing transitions of a state $s = (t, e)$ are independent of e.

Diffuse MRM gradient. The definition in Equation (42) only allows one outgoing transition for each state, which seemed too strict. We prefer a "diffuse" set of outgoing transitions, i.e., still associate a high probability with the transitions which had a probability of 1 due to Equation (42), but also have transitions to neighboring energy classes at the same hour, though less probable the further these energy classes deviate from the original transition target.

This "diffuse MRM gradient" is achieved by applying a generalization to the previous approach. A new parameter k_{MRM} defines over how many potential new transitions the transition probability should be split. For convenience, let us ignore the fact that there is a maximum and a minimum energy class. This problem is handled later on. Let the original transition (according to Equation (42)) lead from (t, e) to (t', e'). We then distribute the transition's probability over all transitions leading from (t, e) to all states $(t', e' + \delta)$ with $\delta \in \{-k_{MRM}, .., k_{MRM}\}$:

$$\sum_{e'=e+\nabla_{MRM}(t)-k_{MRM}}^{e+\nabla_{MRM}(t)+k_{MRM}} p_{(t,e),(t+1,e')}^{(3)} = 1 \tag{43}$$

As to the claim from before, this is indeed a generalization since the strict version of the MRM gradient is achievable by just setting $k_{MRM} = 0$.

We still need to define how this probability distribution exactly works. Similar to the weighted time frame probabilities, we assign weights according to a distance from a "center", which in this cases equates the distance of the transition's target to the transition target we obtain using the strict MRM gradient approach.

For simplicity, we use the same weight function as before, i.e., the transition from (t, e) to $(t + 1, e + \nabla_{MRM}(t) + \delta)$ has weight $k_{MRM} + 1 - |\delta|$.

It remains to treat the cases in which a transition's target state does not exist because it exceeds one of the energy class boundaries e_{min} or e_{max}. As for the strict MRM gradient probabilities, we "bend" these transitions to the nearest state with the same t' and e_{max} or e_{min} respectively. Because there already exists a transition leading to this state, we have to assign the sum of both transitions to it (see Figure 22).

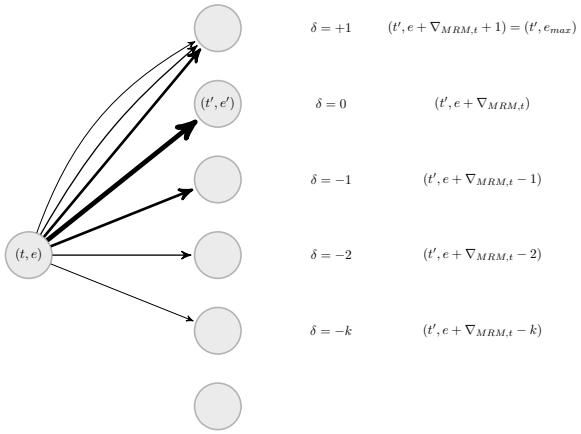


Figure 22: Diffuse MRM gradient for $k = 3$. A thicker edge represents a higher probability of taking the corresponding transition. In this example, (t', e') is the successor of (t, e) according to the MRM gradient. Since $e' + 1 = e_{max}$ and $k = 3$, the probability of $(t, e) \to (t', e_{max})$ is greater than the probability of $(t, e) \to (t', e' - 1)$, indicated by the redirected edges which would otherwise target non-existing states $(t', e_{max} + 1)$ and $(t', e_{max} + 2)$.

The transition from i to j is assigned the weight $p_{ij}^{(3)'}$. Using the notation from Figure 22 this results in a weight calculation as described in Equation (44).

$$p_{(t,e),(t',e')}^{(3)'} = \begin{cases} (k + 1 - |\delta|) & \text{if } e_{min} < e' < e_{max} \wedge t' = t + 1 \\ \sum_{l=\delta}^{k} (k + 1 - |l|) & \text{if } e' = e_{max} \wedge t' = t + 1 \\ \sum_{l=\delta}^{-k} (k + 1 - |l|) & \text{if } e' = e_{min} \wedge t' = t + 1 \\ 0 & \text{otherwise} \end{cases} \tag{44}$$

One can see that δ equals the distance from target state energy class $e + \nabla_{MRM}(t)$ (according to MRM gradient) to e'. Hence, we can rewrite $p_{ij}^{(3)'}$ as in Equation (45).

$$p^{(3)'}_{(t,e),(t',e')} = \begin{cases} k+1-|e'-(e+\nabla_{MRM}(t))| & \text{if } e_{min} < e' < e_{max} \\ & \quad \wedge t' = t+1 \\ \sum_{l=e'-(e+\nabla_{MRM}(t))}^{k} (k+1-|l|) & \text{if } e' = e_{max} \\ & \quad \wedge t' = t+1 \\ \sum_{l=e'-(e+\nabla_{MRM}(t))}^{-k} (k+1-|l|) & \text{if } e' = e_{min} \\ & \quad \wedge t' = t+1 \\ 0 & \text{otherwise} \end{cases} \tag{45}$$

At this point, the weight calculation is defined but the probability cannot be equal to the raw weights since this hurts the condition mentioned earlier in Equation (43). To achieve a total sum of 1 for all outgoing transition probabilities of each state, we can divide the weight of each outgoing transition of state (t, e) by the total sum of weights assigned to all outgoing transitions of that state:

$$p^{(3)}_{ij} = \frac{p^{(3)'}_{ij}}{\text{total weight outgoing from } i} = \frac{p^{(3)'}_{ij}}{(k+1)+2\cdot\sum_{i=1}^{k} i} = \frac{p^{(3)'}_{ij}}{(k+1)^2}. \tag{46}$$

4.3 Resulting models

We model years with $0 \leq t \leq 8784 = 366 \cdot 24$, a single initial state $s_0 = (t = 0, e = 0)$ and a single final state $s_f = (t = 8784, e = 0)$. Thus a model can consist of up to

$$|S|_{max} = \underbrace{1}_{s_0} + 8783 \cdot K + \underbrace{1}_{s_f} \tag{47}$$

many states, which yields a maximum number of transitions equal to

$$|T|_{max} = \underbrace{K}_{\text{outgoing from } s_0} + 8782 \cdot \underbrace{K^2}_{\text{from } K \text{ to } K \text{ energy classes}} + \underbrace{K}_{\text{from left to } s_f} + \underbrace{1}_{\text{self-loop in } s_f}. \tag{48}$$

Of course, these limits assume that all states are reachable considering only transitions with a probability greater than zero when representing probabilities with a precision of, e.g., four digits after the decimal points. Remember that this precision boundary was introduced to reduce the model size and uses the assumption that all transitions removed by it are negligible. For example, for $K = 100$ we already have $|S|_{max} = 878302$ and $|T|_{max} = 87820201$. In the evaluation section (Section 6) we examine the actual model size, which includes only reachable states and only transitions satisfying the precision boundary, depending on different model parameters.

With all parameters described in the previous section, we may construct a great variety of models. The parameters we can tune after the filtering and simulation process are listed in Table 4.

The remainder of this section presents three models (Figure 23) to offer a more intuitive understanding of the probability computation methods. We construct small

Parameter	Description	Range
R	all state rewards are rounded to multiples of R	$\mathbb{N}_0$
K	number of energy (discretization) classes	$\mathbb{N}_{>0}$
prec	maximum number of positions after decimal point for probabilities	$\mathbb{N}_0$
$\xi_p^{(1)}$	weight for probabilities $p^{(1)}$ according to hourly expectancy	0..1
$\xi_p^{(2)}$	weight for probabilities $p^{(2)}$ according to weighted time frame	0..1
$\xi_p^{(3)}$	weight for probabilities $p^{(3)}$ according to diffuse MRM gradient	0..1
Ξ_p	weight vector for probabilities, $\left(\xi_p^{(1)}, \xi_p^{(2)}, \xi_p^{(3)}\right)$	
$\xi_r^{(1)}$	weight for rewards according to hourly expectancy	0..1
$\xi_r^{(2)}$	weight for rewards according to weighted time frame	0..1
$\xi_r^{(3)}$	weight for rewards according to diffuse MRM gradient	0..1
Ξ_r	weight vector for rewards, $\left(\xi_r^{(1)}, \xi_r^{(2)}, \xi_r^{(3)}\right)$	
k_{TF}	time frame size	$\mathbb{N}_0$
k_{MRMG}	k for diffuse MRM gradient	$\mathbb{N}_0$

Table 4: Model parameters.

models with only 20 energy discretization classes to stress the differences. The figure shows two views on each model per row, depicting (1) three days and (2) a single day respectively. The left hand side figures (Figures a, c, and e) all show the same three days of the year and the right hand side figures (Figures b, d, and f) show the transitions corresponding to the same day of the year. All lines are representing a transition in the model where a darker line means a higher probability. This also means that white reflects a transition with the probability zero or thereby the absence of a transition. When there are no ingoing transitions to a state, this state is not reachable and thus removed before model checking.

As intended, using only the probabilities according to the hourly expectancies yields the model with the fewest states and transitions. In Figure 23b, we can also observe that if only a few data points lie below the typical daily curve, the corresponding states with small e values have a lower probability of being reached than the states with the same t but a higher e.

However, this does obviously not work as well if there are not enough data points with such a t and a reasonably high energy value, e.g., due to general lack of data for a given day and year. This is the reason why we introduced the time frame probabilities. Compared to the hourly expectancy probabilities in Figure 23a, Figure 23c presents decidedly more transitions. In this example, the time frame size was set to 7, i.e., the transitions in Figure 23d are based on the depicted day (or rather the underlying data of the depicted transitions) itself as well as the data of the previous and following seven days.

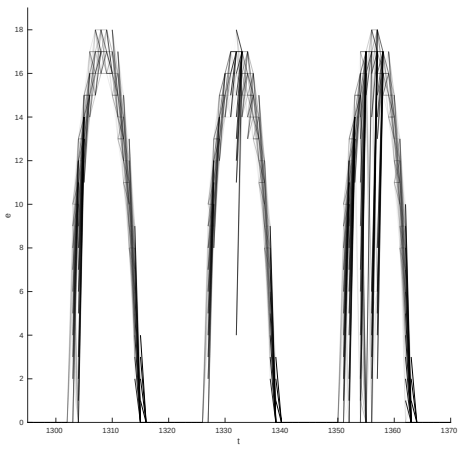
As a result, almost every state under the "arc" described in Figure 23b appears to be reachable while only few days above said arc get included. Although these low energy states can be reached, the ingoing transition probabilities are mostly low-valued (many

are hard to identify by eye). Furthermore, the targets of the most probable outgoing transitions lie on the course of a smoothed version of the mentioned arc, which is why we see the use of time frame probabilities as smoothing.

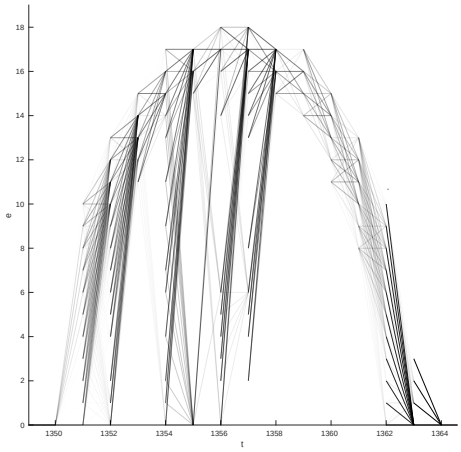
The remaining figures show the probabilities according to the MRM gradient, here with $k_{MRMG} = 10$. Morning and evening are especially apparent in Figure 23e: during the morning hours the probabilities of transitions leading to states with high e values are more probable (darker) than those of transitions leading to states with low e values, conversely so during the evening.

The most obvious difference to the previous two probability measures is the reachability of all states with transitions in the depicted t and e range. As can be seen in Figure 32b later in the evaluation section, this amounts to rendering 99.92% of all states reachable when using only the diffuse MRM approach for 500 energy classes, while the hourly expectancy approach and weighted time frame approach render only about 10% and 33% reachable. This could be changed, of course, but the primary aim of the MRM gradient is to strengthen the probabilities derived from $p^{(1)}$ or $p^{(2)}$ which correspond to a "typical day", while this "typical day" is not derived from measured data but from the ideal DNI values. With a sufficiently small $\xi_p^{(3)}$ and probability precision as well as a higher value for K, we should be able to reduce the number of unnecessary states during the nights. This effect can be seen in Figure 32a as compared to Figure 30a.

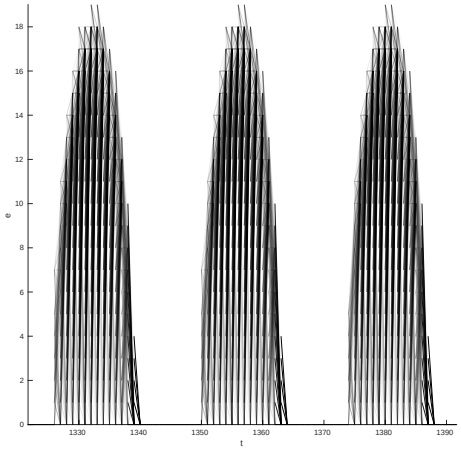
Section 6 will present resulting model dimensions after Section 5 presents the implementation of our ideas.



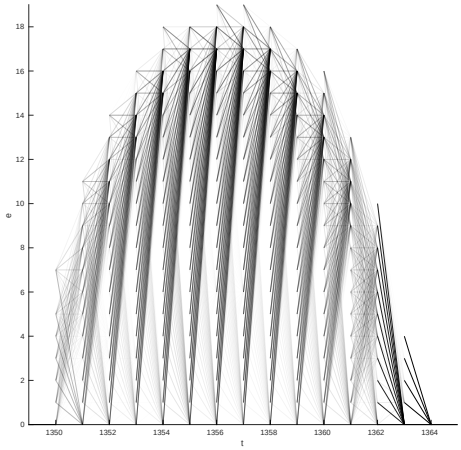
(a) $\Xi_p = (1, 0, 0)$, three days.



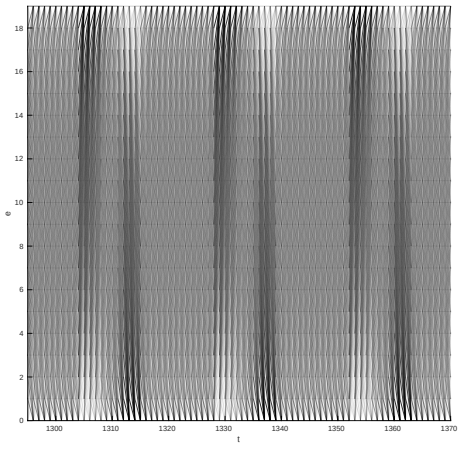
(b) $\Xi_p = (1, 0, 0)$, one day.



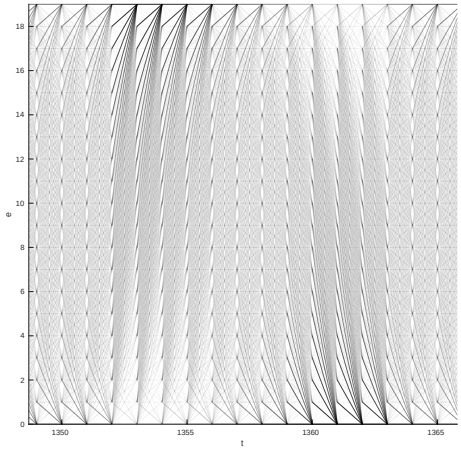
(c) $\Xi_p = (0, 1, 0)$, three days.



(d) $\Xi_p = (0, 1, 0)$, one day.



(e) $\Xi_p = (0, 0, 1)$, three days.



(f) $\Xi_p = (0, 0, 1)$, one day.

Figure 23: Resulting model transitions for the different probability computation methods for $K = 20$. Lines represent transitions, a darker line means a higher associated probability.

5 Implementation

This section covers the implementation of the data preparation and model construction. Section 5.1 presents the layout of the underlying software chain, briefly discussing the role of each component involved. After this overview, Section 5.2 explains details of the filtering algorithm and Section 5.3 presents the settings which are necessary in order to utilize the simulation with the results of the filter for the weather data. In Section 5.4, details of the Model Builder are explained.

5.1 Software chain

The main concept of our implementation is closely related to the strategy pattern. Every core concept of the process from MRM computation to the construction of a model is encapsulated in a dedicated class with its own input and output files (see Figure 24). Although we eventually reduced the number of concrete strategies per abstract strategy to exactly one, we still benefit from this approach as, e.g., for experiments, the model builder can be invoked multiple times without us having to re-run the MRM computations, weather filtering, sundata file generation, and solar tower simulation every single time.

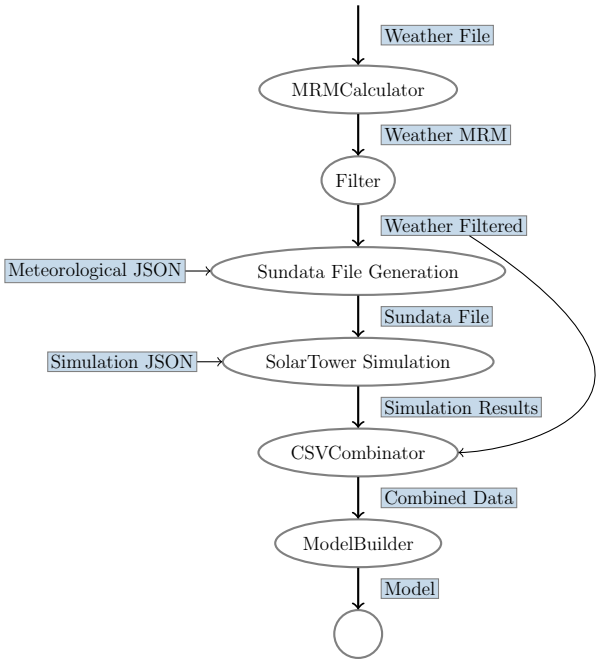


Figure 24: Software chain and important (intermediate) files.

The caveat is obviously that a single run takes longer due to the need for serial reading and writing, but we realized that the filtering process and consequently also the sundata file creation should ideally only be performed once. Afterwards, a solar tower can be simulated and examined using multiple models. This reasoning and the

weatherHEADLESS.csv	
Date	
GHI	W/m²
DNI	W/m²
DHI	W/m²
DHI shadowband	W/m²
UVA	W/m²
UVB	W/m²
Air temperature	°C
Barometric pressure	hPa
Relative humidity	%
Wind speed	m/s
Wind direction	deg
Standard deviation of wind direction	deg

weatherMRM.csv	
Date	
GHI	W/m²
DNI	W/m²
DHI	W/m²
Azimuth	deg
Zenith	deg
MRM	W/m²

weatherFiltered.csv	
Date	
DNI	W/m²
Azimuth	deg
Zenith	deg
MRM boundary	W/m²
MRM	W/m²

simulationEnergy.csv	
Date*	
Energy	Wh/a

combinedData.csv	
Date	
DNI	W/m²
Azimuth	deg
Zenith	deg
Energy	Wh/a
MRM	W/m²

Table 5: CSV files. The date is always given in the format `YYYY-MM-DD HH:mm`. For (*) see Section 5.3.

fact that a detailed simulation takes much longer than the model construction leads to the proposed ordering of the steps as opposed to modeling weather data and simulating each weather model.

The extra space needed for the intermediate files is negligible considering that the weather file with MRM data (`weatherMRM.csv`) is our largest CSV and is only 226 MB in size as compared to the 1.1 GB sundata file or the model files which can grow to a virtually infinite size: for the evaluation we created model files with more than 20 GB of content for just the transitions, one transition (two numerals) per line in a pure text file.

The names and structures of the CSV files are listed in Table 5.

Before we can start the execution of our program, the weather file downloaded from the SAURAN page must be prepared as it contains unreadable content after the last entry which makes up approximately 40% to 50%[5] of the file size. During this step we also remove the two header lines and save the result as `weatherHEADLESS.csv`.

In a first step, our program calculates the MRM values for each hour and the solar position in azimuth and zenith for each minute in the weather file and saves the results in a new `weatherMRM.csv` file. Minute data in between full hours gets an MRM value of -1 to indicate that no MRM was calculated for that time. This facilitates the boundary computations later on, during which we iterate over the MRM values.

Using the `weatherMRM.csv`, the `Filter` can compute the MRM boundaries for the weather data and finally remove invalid entries. The resulting `weatherFiltered.csv` contains the date strings, DNI values, solar positions, MRM boundaries, and MRM values. Note that we only keep the MRM boundaries in the filtered weather file to

[5] 13 MB of a 32 MB file and 62 MB of a 128 MB file

enable us to plot and verify the results.

With the solar positions and DNI values we can create a sundata file for use in the solar tower simulation which creates the `simulationEnergy.csv`. The `CSVCombinator` can then append the simulation energy values to the filtered weather data, which yields the `combinedData.csv`, the basis of the `Model Builder`.

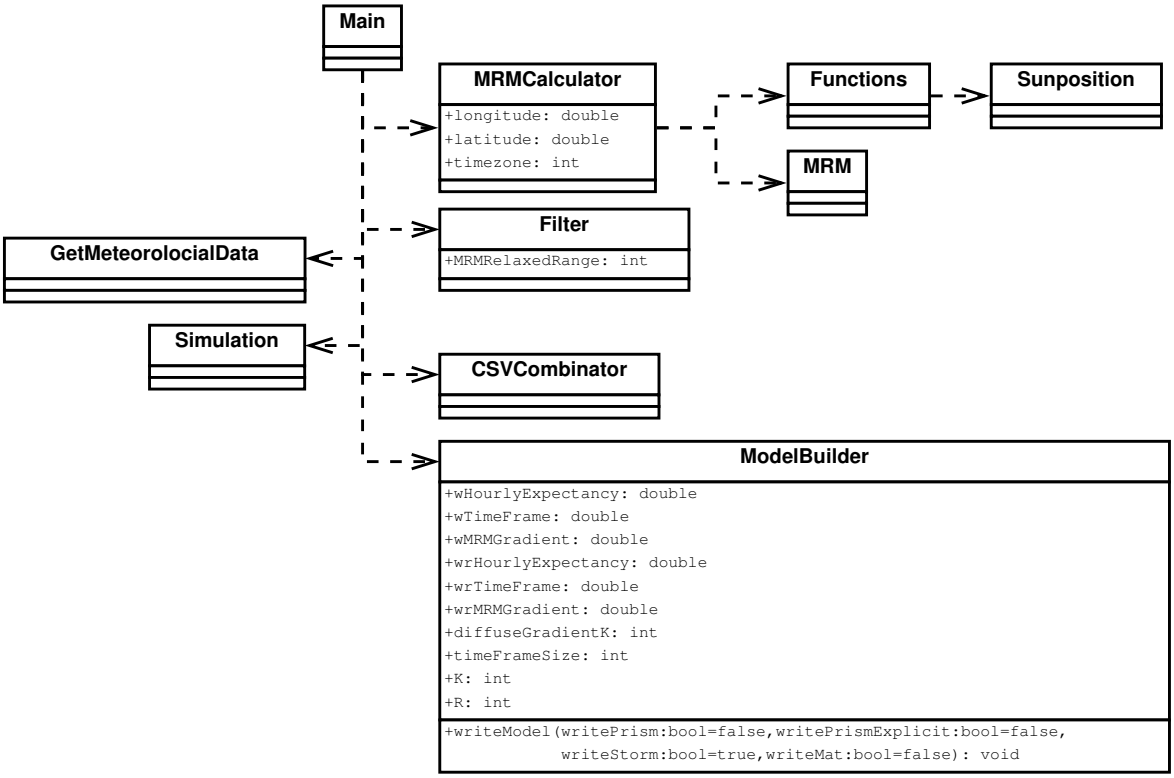


Figure 25: Simplified UML diagram.

Figure 25 shows a UML diagram of the most important components of the implementation. Note, that `MRM` is a modification of the MRM source code provided by Dr. Kambezidis [14], and `Sunposition`, `GetMeteorologicalData`, and `Simulation` are part of the solar tower simulation.

All paths, file names, and settings can be modified within the `Main` (before the actual execution code) or via command line parameters (see `./Simulation -h`).

5.2 Filtering the weather data

As described in Section 3.2, we need to compute the MRM for the construction of upper boundaries for the filter. Mr. Kambezidis kindly provided the Fortran source code of the latest MRM version. Our own attempt of implementing this radiation model using only the literature failed, most likely due to the differences between the literature values and the those used in the code[6].

[6]for details on this see Section 2.3.

The code was then translated into C++ and needed the following changes to better fit our approach:

- remove need for external configuration file,
- embed in OO approach, use as function instead of individual main executable,
- remove main loop and need for input file, pass everything via function parameters, and
- remove output of result file, return value instead.

Originally, the MRM code reads a configuration file containing the location as longitude, latitude, standard latitude, timezone, height above normal zero, etc., as well as a list of dates with a time of day, corresponding relative humidity, air pressure, etc. What we need, however, is a component which simply tells us for a given date and location, what the MRM value is. We do not want to rewrite the given weather data file or have extra configuration files.

Hence, we wrote a class `MRMCalculator` which iterates over the original weather data and calls the MRM calculation for each entry, returning the calculated value directly instead of first having to write and read an additional output file.

5.3 Solar tower simulation

In order to simulate the individual minutes remaining in the filtered weather data file, we utilize the solar tower simulation with its executable in `$SIMPATH`.

Unfortunately, the solar tower simulation does no longer support weather data files in CSV format, so we first need to convert them into so called sundata files. The conversion is done by the executable `GetMeteorologicalData` in `$SIMPATH/bin` after invoking `make GetMeteorologicalData` in `$SIMPATH/src/service`.

Listing 1: Generation of a sundata file from the filtered weather data

```
./bin/GetMeteorologicalData -s csv
                            -c weatherFiltered.csv
                            -j meteorological.json
                            -o src/service/sunfile.sun
```

Afterwards, the solar tower simulation can be run by invoking its executable with command line parameters as follows:

Listing 2: Running the solar tower simulation

```
./Simulation -s
             -i simulation_master_thesis.json
             -c ../solar\ tower/data/helio100/positions.csv
```

This tells the Simulation to simulate a heliostat field layout or configuration as given in `positions.csv` and configures the simulation (selection of simulation type (solar

tower or offshore wind), location of sundata file, receiver settings, etc.) according to the `simulation_master_thesis.json`.

In this settings file we also select to use our previously calculated zenith and azimuth values, since at this point we already know these values and can omit having to synchronize the location and timezone values between the MRM settings and the simulation settings.

For this thesis, we used the simulation as of Subversion (SVN) revision 542, as this is the latest version we tested which produced sensible results. Among other changes, the simulation now reorders the sundata entries which we believe may be the cause of unsatisfactory results: energy curves of a single day are stretched to multiple days and depending on the year of the weather data the curves are shifted to the left or right. The issue is under investigation and since revision 542 works, we could eventually continue.

The simulation's output can be found in `$SIMPATH/momentenergy`, which used to contain a date string followed by the energy received during the given minute. We now need to merge the energy file and the weather file, so that we can use both real energy and the MRM values for probability computation. The initial approach was to search both files for matching dates. This way, one could simulate a single year and relate it to the weather file and if a simulation value should be erroneous or missing, e.g., -nan, we could still use all other values. However, the simulation no longer outputs the date string, so we now have to copy the date column from the weather data file `weatherFiltered.csv` to the energy file `simulationEnergy.csv` before combining them.

5.4 Model construction

The class `ModelBuilder` is responsible for the construction of the DTMC models. It reads the `combinedData.csv` written by the `CSVCombinator` which contains all dates, simulation energy, and MRM values. While reading it computes the maximum energy and MRM values so it can set the respective discretization classes accordingly. When the entire file is read, we can assign each entry to a state. To this end we introduce two maps, one named `cntMap` which contains for a key (t, e) the number of entries within the respective state, and a second map `cntHPtsMap` with the number of "hour points" for a given t. Additionally, while we are traversing all entries, the hourly averages of MRM and energy are being computed for each state.

In the next step the different probability measures are computed in parallel OpenMP sections and afterwards added with the corresponding probability weights. Each probability measure is returned as a three dimensional matrix (vector of vectors of vectors of type float) in the form `p[t][j][k] = v` which stands for "the probability of going from state (t, j) to $(t + 1, k)$ is v". Note that this format can represent all transitions except for the self-loop in the final state. During the computation of the probabilities, we have to divide by the number of entries at a given t, i.e., by `cntHPtsMap[t]`. If this is zero, we choose to "hold the energy value", which means that we set the probability of `p[t][j][j] = 1`. An alternative solution would be to set "pull missing data to

zero", `p[t][j][0] = 1`, but we think that it is more realistic to assume that missing data for one hour means a (hopefully) short outage of the measuring systems and that the modeled energy should be in the vicinity of the previous hour.

A second decision concerns the numeric precision of the probabilities. As mentioned in Section 4, we want to restrict the representation of each probability to a given number of digits after the decimal point. Hence, we trim all probabilities but still have to assure that the sum of probabilities of all outgoing transitions of each state equals 1. In an "optimistic" approach, the proposed solution adds the difference to 1 to the probability of the transition which, after trimming, leads to the highest reachable energy class.

During the computation of the time frame probabilities, we also calculate the time frame rewards of each state as this requires the exact same loops and iterations. Thus, if the user wants to create a model with $\xi_r^{(2)} > 0$, $\xi_p^{(2)}$ has to be greater than zero as well. The remaining rewards, hourly expectancy and MRM average, are already computed during `readData`. So it only remains to compute the final state rewards as the weighted sum of the mentioned partial rewards.

5.5 Exporting the model

The model can now be written in the input format of a DTMC model checker. We initially wanted to utilize PRISM to compute the average annual energy reception at the receiver of the Helio100. To this end, we used the PRISM language to write a single `.prism` file which is easily readable (good for finding errors) and can be parsed, built, and evaluated by PRISM.

But this approach proved very slow. Even after modifying transitions `[] t=1 & e=3 -> 0.3:(t'=t+1) & (e'=0) ...` to look like `[] t=1 & e=3 -> 0.3:(t'=2) & (e'=0) ...` and using the explicit engine, a model with $K = 100$ took about 21 hours to parse and build.

Hence, we decided to write the explicit representation directly. The explicit format takes multiple files dedicated to either the transitions, states, labels, etc. instead of just one file for all information. Additionally, states can no longer be identified by the values of their variables but only by their IDs. In our case, the DTMC must be represented by a state file `model.sta`, a transition file `model.tra`, a label file `model.lab` defining the initial state, and a state reward file `model.srew`.

Due to the simple general layout of our models we can compute the set of reachable states by looping over the hours of the year and energy classes. A state $s' = (t', e')$ is reachable if and only if there is a transition probability greater than zero from a reachable state $s = (t' - 1, e)$ to s', for any e, or if s' is the initial state. So for every t, we memorize the indices t' and e' of the states which are reachable from any state (t, e) and only regard those reachable states in the next iteration. During this loop we also enumerate the states and therewith compute the unique state IDs.

5.5.1 Choosing a model checker

Although we verified our own explicit files with the explicit files that can be exported by PRISM, using the explicit format resulted in different values than using the implicit format, in particular, the average energy was always zero. It turns out that PRISM can use the implicit model and export the explicit model including state rewards, but it does only import a partial model: states, transitions, and labels are imported while state rewards are ignored.

Thus, if we were to use PRISM, we could not skip the unnecessary[7] building phase of the models. So we needed to switch the model checker. The initial idea was to use the Markov Reward Model Checker (MRMC), but using this model checker we cannot simply inquire what the average yearly energy reception at the solar tower is, since for our use-case, it can only answer the question whether the expected reward lies within a given interval. We would thus have to implement a search algorithm which narrows down this interval until it only contains a single number. However, STORM, a successor to MRMC, fulfills all requirements presented so far: it can import an explicit model format and output the average annual energy of our models.

In the following sections, we present the three implemented solutions and pitfalls we encountered: (1) implicit and (2) explicit PRISM models as well as (3) explicit STORM models.

5.5.2 PRISM

If PRISM is to be used, it is possible to run into the following error:

```
Exception in thread "main" java.lang.OutOfMemoryError:  Java heap space
```

On a machine with sufficiently large main memory this can be fixed by increasing the heap size. To do so, check for the maximum heap size available to the Java virtual machine (VM) via

```
java -XX:+PrintFlagsFinal -version | grep HeapSize
```

and set this to an appropriate value. During this thesis, a maximum heap size (flag `-Xmx<SIZE>`) of 300 GB (!) sufficed for models with $K = 100$. Smaller values might suffice but this requires lengthy testing and is not the solution we chose. To avoid the need to write a wrapper in order to invoke `java -Xmx300g wrapper`, we had to set the Java option for all sessions:

```
export _JAVA_OPTIONS="-Xmx300g".
```

Now the Java VM may allocate a larger heap, but PRISM does not use this by default. However, there are two options available: `cuddmaxmem` and `javamaxmem`.

This leads to the following call:

[7] from our perspective as we can already export the explicit model format

Listing 3: Prism call: Build model

```
prism ${MODELNAME}.prism -explicit
                         -cuddmaxmem 4g -javamaxmem 300g
                         ${PROPFILE}
```

Using the explicit PRISM format, we have to specify the model type (dtmc) and can invoke the model checker as follows:

Listing 4: Prism call: Import model and check for properties

```
prism -importmodel ${MODELNAME}.all
      -dtmc
      -exportresults log
      ${PROPFILE}
```

for a file `${PROPFILE}` containing the properties which the model is to be checked for.

5.5.3 Storm

The STORM file format is similar to the MRMC format, which PRISM can also export via the switch `-exportmrmc`. The transition files begin with the specification of the model type, `dtmc` in our case, and follow with the list of transitions – one transition per line – as `sID1 sID2 prob`, which stands for the probability of going from the state with ID "`sID1`" to the state with ID "`sID2`" is `prob`, where state IDs start at 0.

Listing (5) Transitions

```
dtmc
0 1 1
1 2 1
. . .
29 30 0.6541
29 31 0.0125
. . .
58610 68610 1
```

Listing (6) Labels

```
#DECLARATION
init
#END
0 init
```

Listing (7) State rewards

```
30 1740
. . .
80 517440
81 886740
. . .
```

Figure 26: STORM model files.

Each state can be assigned a label in the `.lab` file. In this thesis, the file is close to empty as only the initial state has to appear.

The state reward file lists for each state ID the corresponding state reward, one per line. We omit the state rewards equal to zero.

5.6 Using the model

In order to utilize the model, we need to define the properties of interest, which can be done in a property file. The property file used in this thesis can be seen in listing 8 and is named `average.prop`.

Listing 8: Property for average reward

```
R=? [ C<=8784 ]
```

Checking for this reward property yields the cumulative reward which is achieved after 8784 time steps which, for our models, means the desired average annual energy when regarding the initial state.

We can then invoke STORM via

Listing 9: Storm call

```
./storm --explicit ${MODELNAME}.tra ${MODELNAME}.lab
        --staterew ${MODELNAME}.srew
        --prop ${PROPFILE}
```

to evaluate the models.

6 Evaluation

In this section, we examine the resulting models based on model sizes, computation duration, and of course the estimated average annual energy reception.

6.1 Preparations

All experiments were undertaken on blade21 of the Ultra High-Speed Mobile Information and Communication (UMIC) cluster, `blade21.informatik.rwth-aachen.de`. Time measurements were done using `time` for STORM and `ctime` for our own code. We refer to those times as STORM time and building time, where the building time includes the necessary steps from reading the `combinedData.csv` to writing the model files, including both steps.

Listing 10 shows a script we used to build and evaluate various models. When the script finishes one model, it copies the relevant log files into a single file, deletes the model files (which required up to approximately 22 GB for the transition files) and starts the process for the next model. Depending on the model sizes, multiple scripts could be executed in parallel.

Listing 10: Storm script

```
#!/bin/bash

declare -a arrk=(50 100 500)
declare -a arrwp=("1 0 0" "0 1 0" "0 0 1" "0.6 0.3 0.1" )

wr="1 0 0"

HOME=~
PROPFILE="${HOME}/average.prop"

for k in "${arrk[@]}"; do
    for wp in "${arrwp[@]}"; do

        MODELNAME="${wp// /}${wr// /}${k}"
        BUILDERLOG="${MODELNAME}.builder.log"
        TIMELOG="${MODELNAME}.time.log"
        STORMLOG="${MODELNAME}.log"

        STORMCMD="$HOME/storm --explicit ${MODELNAME}.tra ${MODELNAME}.
            ↪ lab --staterew ${MODELNAME}.srew --prop ${PROPFILE}"

        echo -wp $wp -K $k -wr $wr
        (./Simulation -wp $wp -K $k -wr $wr -mn ${MODELNAME}) >>
            ↪ $BUILDERLOG 2>&1;

        echo "starting storm"
        { time $STORMCMD > $STORMLOG; } 2> $TIMELOG

        cat $TIMELOG >> $STORMLOG
```

```
            cat $BUILDERLOG >> $STORMLOG

            rm $BUILDERLOG
            rm $TIMELOG

            rm ${MODELNAME}.tra
            rm ${MODELNAME}.lab
            rm ${MODELNAME}.srew

        done;
done;
```

For reference, we computed the maximum and average received energy per year, 18.45 $^{\text{MWh}}/_{\text{a}}$ and 10.06 $^{\text{MWh}}/_{\text{a}}$, respectively. These values were calculated from the `combinedData.csv` file by summing up the minute-wise maximum or average of the simulated energy values. We consider values close to the average as realistic (read as "good") values.

6.2 Time frame size

Figures 27 and 28 depict the influence of the time frame size on models. To get a clear insight, we chose to set $\xi_p^{(2)} = 1$, i.e., derive the transition probabilities only from the time frame approach and use models of "medium size" with 100 energy discretization classes. Remember that k_{TF} is the number of days to either side of the current day which are to be considered for the probability construction of the transitions representing that day. Thus, we inspected the difference between regarding three ($k_{TF} = 1$) and two weeks ($k_{TF} = 7$), approximately two months ($k_{TF} = 30$), and one year ($k_{TF} = \frac{366}{2} = 183$).

For the given weather files and samples, we can see that the function annual energy over time frame size resembles an exponential decay until k_{TF}=183 for $\Xi_r = (0, 1, 0)$, while both the model size and computation time for the model checker show a logarithmic increase. The energy decrease is due to the fact that for the given location the winter is longer than the summer. Hence a time frame which is closer to spanning the entire year is under a greater influence of the low-valued winter days than high-valued summer days. As can be seen in the plot, using a k_{TF} value of 7 up to 30 yields good results, i.e., energy values close to the computed average of the simulation energy file.

Regarding only $k_{TF} \in \{1, \cdots, 15\}$, we can see in Figure 28 the number of states increases by approximately 23%, from 284870 to 351116, and the number of transitions increases by 50%, from 18 million to 27 million. However, the time we spend for the computation of those models only increases by 29% (169 seconds and 218 seconds, respectively). Additionally, the models only take up 435380 kB ($k_{TF} = 15$) of main memory during the STORM computations. Consequently, the time frame size can be chosen freely within the interval bounded by 1 and 15.

The only downside the largest of these values entails is that the time spent during the STORM evaluation also increases by 50% which means an increase by 53 minutes. But this issue will be discussed later.

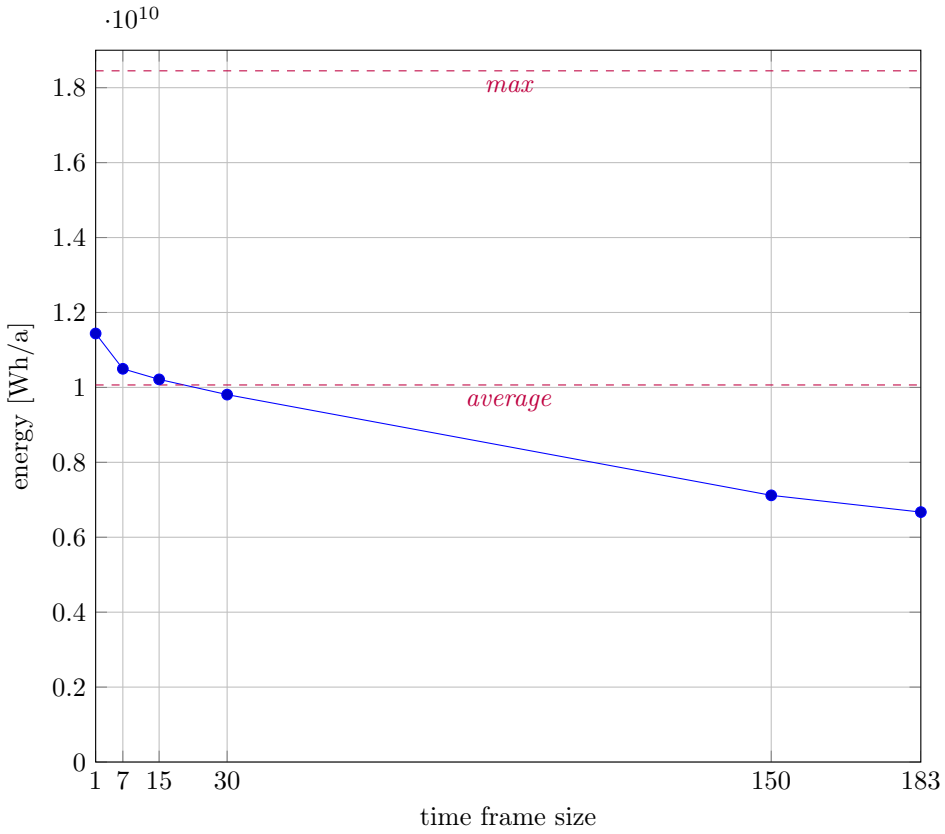


Figure 27: Influence of the time frame size on the average annual energy. Variable k_{TF}, $\Xi_p = (0, 1, 0)$, $\Xi_r = (1, 0, 0)$, prec=4, K=100, R=1.

As a consequence of our findings, we advise to use $k_{TF} = 7$ as this achieves good results and still uses fewer states and transitions than a larger and possibly worse time frame size. Furthermore, a time frame of effectively two weeks is intuitively more robust than one of just three days when considering measurement outages. Contrariwise, the seasonal weather deviations should be negligible for $k_{TF} = 7$ as compared to time frames spanning over one or even two months.

6.3 Reward weights

Figure 29 displays the differences between the three implemented reward structures by presenting the average yearly energy output STORM computes when a given model contains only state rewards in accordance with either of the reward structures or a weighted mixture.

The MRM reward is lower than the other rewards by an order of magnitude, which is to be expected since the MRM reward does not reflect energy but only DNI values. Should the rewards computed by the remaining reward structures prove to be too high, the MRM rewards can be utilized to decrease the overall reward. E.g., the yearly average energy for $\Xi_r = (0.6, 0.3, 0.1)$ is $1.03 \cdot 10^{10}$ as opposed to $1.11 \cdot 10^{10}$ for

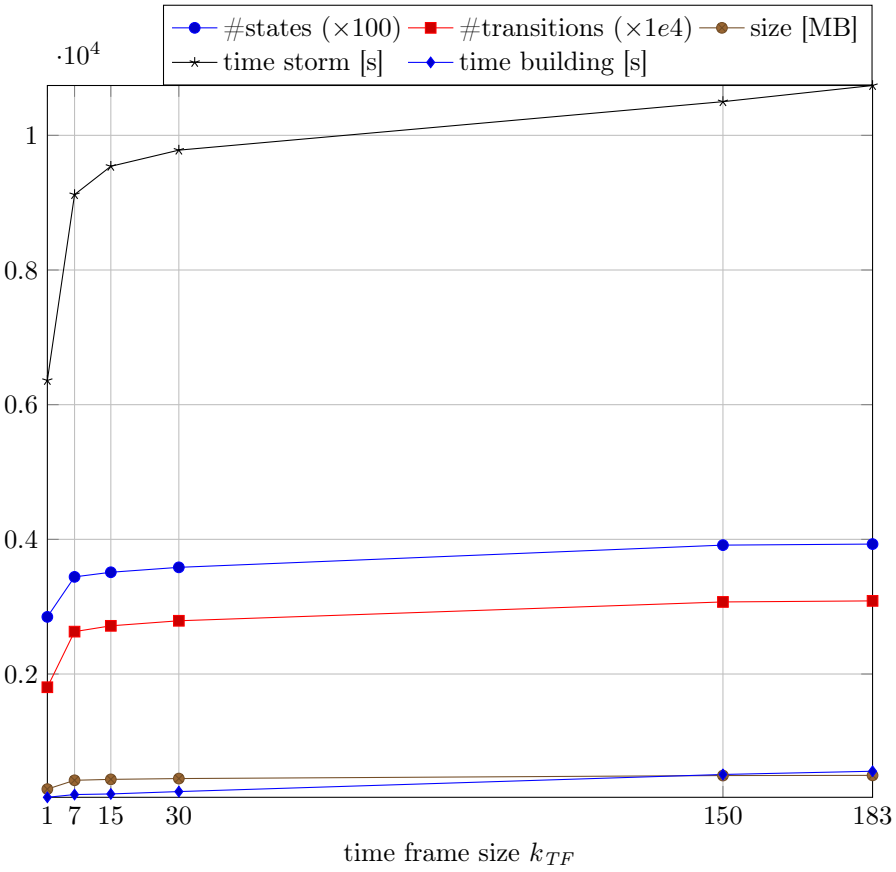


Figure 28: Influence of the time frame size on the model size and computation duration. Variable k_{TF}, $\Xi_p = (0, 1, 0)$, $\Xi_r = (1, 0, 0)$, prec=4, K=100, R=1.

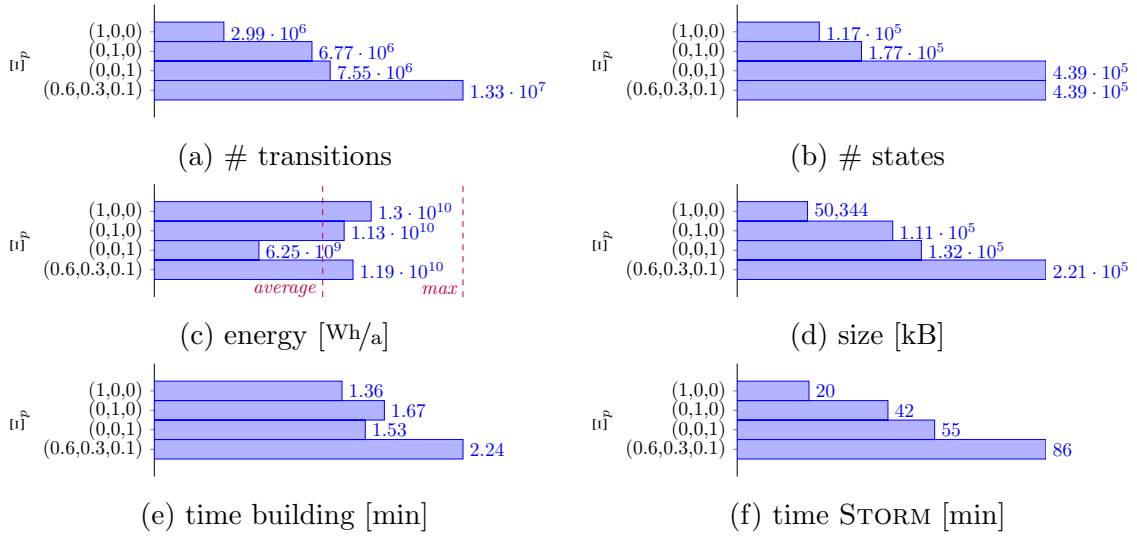
Figure 29: Influence of reward weights on a model with $K = 100$, 877807 states, and 43464337 transitions, derived with $\Xi_p = (0.6, 0.3, 0.1)$, $k_{TF} = 7$, $k_{MRMG} = 10$.

$\Xi_r = (0.6, 0.4, 0)$.

6.4 Number of energy discretization classes

Figures 30 through 32 show the available measures (number of states, number of transitions, model size during STORM computation, average annual energy production, STORM time, and building time) for four different models per number of energy classes K, where the latter is in $\{50, 100, 500\}$. For each K, the models are created with

the same settings except for changing probability weights stressing the accumulated differences between the approaches (and a weighted mixture).



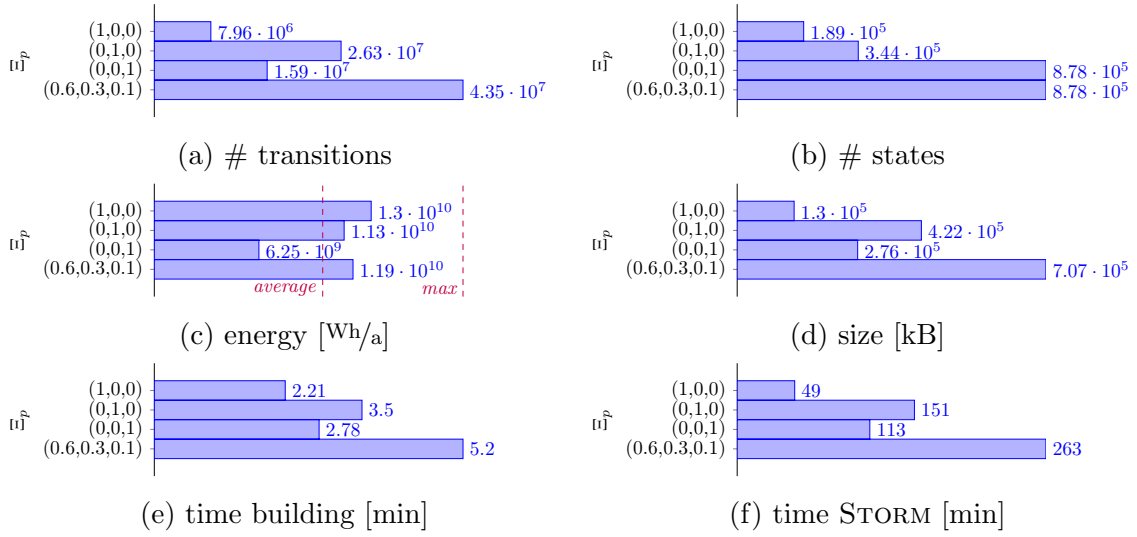
(a) # transitions (b) # states

(c) energy [Wh/a] (d) size [kB]

(e) time building [min] (f) time STORM [min]

Figure 30: Results for $K = 50$, $R = 1$, $\Xi_r = (1, 0, 0)$, prec=4, k_{TF}=7, k_{MRMG}=10.

We observe that with an increasing accuracy of the model, the average annual energy decreases for all models with $\xi_p^{(1)} < 1$ but stays within the confines of "good" values for $\Xi_p = (0, 1, 0)$ until $K = 100$ and even until $K = 500$ for the chosen mixed probabilities, $\Xi_p = (0.6, 0.3, 0.1)$. Remember that the "good" value range is not purely objective since it too relies on the measured weather data with noise and missing data points and we can only consider the accumulated annual average in this thesis.



(a) # transitions (b) # states

(c) energy [Wh/a] (d) size [kB]

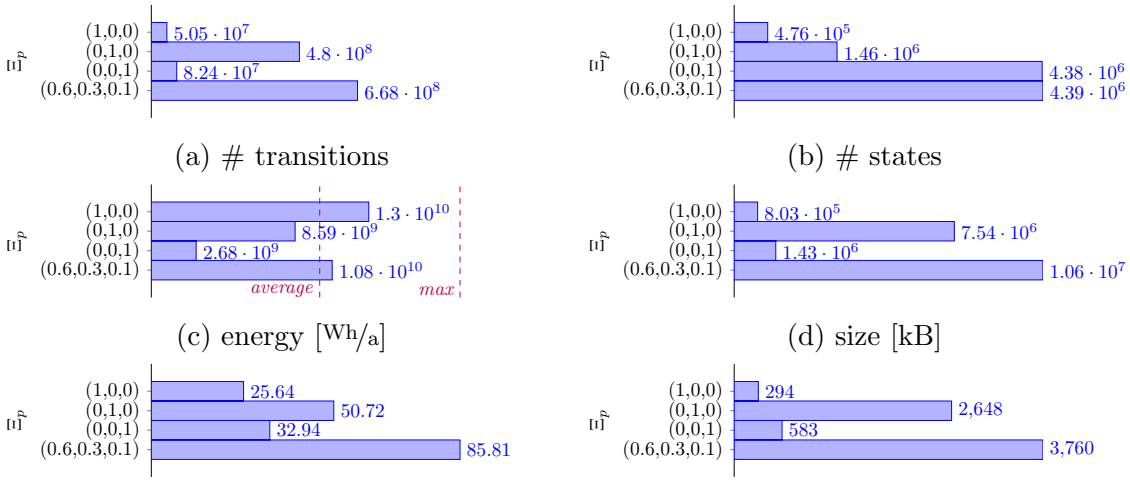
(e) time building [min] (f) time STORM [min]

Figure 31: Results for $K = 100$, $R = 1$, $\Xi_r = (1, 0, 0)$, prec=4, k_{TF}=7, k_{MRMG}=10.

Focusing on the results shown in Figure 32, we can observe the importance of feasible probability weights as they greatly influence not only the average energy but also

the size of the resulting models. The intuition behind this could already be seen in Figure 23, but we can now measure the actual scale: in our four models for $K = 500$, the number of states varies between 0.47 million and 4.39 million, while the number of transitions varies between 50.46 million and 668 million.

We can also see that the number of transitions is not a function of the number of states, as the model according to the MRM gradient features 4.38 million states, only 0.01 million less than the model with mixed probability measures, but consists of only 82 million transitions, 586 million fewer transitions than the mixed model, and interestingly 398 million fewer transitions than a model with only a third of its states ($\Xi_p = (0, 1, 0)$).



(a) # transitions (b) # states

(c) energy [Wh/a] (d) size [kB]

(e) time building [min] (f) time STORM [min]

Figure 32: Results for $K = 500$, $R = 1$, $\Xi_r = (1, 0, 0)$, prec=4, k_{TF}=7, k_{MRMG}=10.

The time spent during the computation of the models is well below two hours (86 minutes) even for the largest model for which we computed all proposed probability measures. Notice that while the STORM evaluation took more than ten times the evaluation time for $K = 500$ and $\Xi_p = (0.6, 0.3, 0.1)$ compared to $K = 500$ and $\Xi_p = (1, 0, 0)$, the building time only increased by a factor smaller than four, due to the implemented parallelization of the probability computations. Under the aspect that the theoretical size of this model is defined by $|S|_{max} = 4\,391\,502$ and $|T|_{max} = 2\,195\,501\,001$, and that the actual size is given by $|S| = 4\,388\,111$ and $|T| = 668\,095\,969$, we consider this sufficiently fast. Note that the building time already includes the time spent for writing the output files.

The problematic time aspect is the time spent for model checking. For a model with this many transitions, STORM needed more than two days' time (3760 minutes $\approx$ 2.6111 days).

7 Conclusion

In this final section, we draw conclusions from the results and challenges of this thesis and present possible augmentations in Section 7.1.

The initial task was the creation of a software chain which allows the creation of probabilistic solar tower models from weather data files with partially erroneous entries. The proposed solution is the software chain depicted in Figure 24, which allows future users to exchange parts of it, such as the MRM calculation, the filter, or the solar tower simulation without a big overhead.

Afterwards, the first problem is the creation of a filter for erroneous weather data, which we solved with the help of the MRM model, finding fitting parameter values, and applying relaxations and additional cut-offs to get daily upper boundaries and eliminate false readings during night time. The quality of this filter is especially visible in Figure 14, which also shows that our filtering approach reflects seasonal weather fluctuations.

Using the proposed modeling tools, we can create a great variety of probabilistic models of solar tower power plants. We have shown that these models are analyzable by model checkers and yield realistic values for the average annual energy reception even though we could only operate on less than six years of data. We have also seen that the creation of those models is possible in agreeable time which renders the work of this thesis a feasible approach for the investigation of possible solar tower power plant configurations.

The use of probabilistic models in this context does not stop at the calculation of an annual average energy but they allow for the formulation and analysis of far more intricate properties. Using one of our models, it is possible to inquire how probable it is to reach a critically high or low energy value during any given period of time, e.g., on weekends or New Year's Eve, for which the energy demand is especially high or low.

7.1 Future work

The biggest flaws are memory limitations and the fact that existing model checkers, to the best of our knowledge, do not detect the simple nature of our models and thus take too long for the calculation of the average annual energy. We thus propose to write a dedicated piece of software for checking properties on the utilized subset of DTMCs. This simplified model checker could omit checking for loops, as there are none in our models except for the self-loop in the final state. Additionally, we already assert that the sum of all outgoing transitions of a state is 1 and we can also guarantee that there are no deadlocks. In comparison with an implicit model checker we can also muster the advantage that no guards need checking and that we enumerate the states in ascending order in accordance with the time, so that from a state (t, e) at most K states must be considered possible successor states.

For simpler property formulations, it is easily possible for the models to be fitted with appropriate labels such as "day", "summer", or "holiday" based on the hour of the year alone, or labels such as "critical" for states modeling an hour of the year with

high energy demand but a low energy reception. Note that the proposed labels do not have to be (although they can be) manually retrofitted to the state space, but that this is possible computationally during a single loop over all reachable states (which we already pre-compute) and a condition such as $t \equiv 8 \; mod \; 24 \wedge e < \lfloor 0.1 \times K \rfloor$.

Considering the filter for the weather data, we can imagine that, similar to finding the upper boundaries, it could be useful to define lower boundaries to remove erroneous DNI measurements at or close to 0 $^{W}/_{m^2}$.

Regarding the computation of the transition probabilities in conjunction with possibly missing data, it could be investigated whether it might be better to interpolate instead of holding the last known energy class/reward.

Furthermore, when we compute the diffuse MRM gradient, we need to sum up probabilities of transitions whose original targets would exceed the minimum or maximum energy class (see Figure 22). Instead, we could cut those transitions, i.e., set the corresponding transition probabilities to zero, and scale-up the remaining transition probabilities.

References

[1] Storm. developed by the verification group headed by professor Katoen at RWTH Aachen University.

[2] R.J. Aguiar, M. Collares-Pereira, and J.P. Conde. Simple procedure for generating sequences of daily radiation values using a library of Markov transition matrices. *Solar Energy*, 40(3):269 – 279, 1988. ISSN 0038-092X. doi: http://dx.doi.org/10.1016/0038-092X(88)90049-7. URL `http://www.sciencedirect.com/science/article/pii/0038092X88900497`.

[3] U. Amato, A. Andretta, B. Bartoli, B. Coluzzi, V. Cuomo, F. Fontana, and C. Serio. Markov processes and Fourier analysis as a tool to describe and simulate daily solar irradiance. *Solar Energy*, 37(3):179 – 194, 1986. ISSN 0038-092X. doi: http://dx.doi.org/10.1016/0038-092X(86)90075-7. URL `http://www.sciencedirect.com/science/article/pii/0038092X86900757`.

[4] V. Badescu. *Modeling solar radiation at the Earth's surface: recent advances*. Springer Verlag, 2008.

[5] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008. ISBN 026202649X, 9780262026499.

[6] B. Belhomme, R. Pitz-Paal, P. Schwarzbözl, and S. Ulmer. A fast ray tracing tool for high-precision simulation of heliostat fields. *Solar Energy Engineering-Transactions of the ASME 131*, (3), 2009.

[7] J.A. Cole and J.D.F. Sherriff. Some single- and multi-site models of rainfall within discrete time increments. *Journal of Hydrology*, 17(1–2):97 – 113, 1972. ISSN 0022-1694. doi: http://dx.doi.org/10.1016/0022-1694(72)90068-6. URL `http://www.sciencedirect.com/science/article/pii/0022169472900686`.

[8] DLR. Solarthermische Kraftwerke sind zuverlässige Technologie für die Energiewende. Website, 2011. URL `http://www.dlr.de/dlr/presse/desktopdefault.aspx/tabid-10172/213_read-1898/#/gallery/4252`. last visited on March 28, 2016.

[9] DLR. Energy research. Website, 2011. URL `http://www.dlr.de/dlr/en/desktopdefault.aspx/tabid-10198/444_read-260/#/gallery/88`. last visited on March 28, 2016.

[10] Carmen-Ana Domínguez-Bravo, Pascal Richter, Gregor Heiming, Sebastian-James Bode, Emilio Carrizosa, Enrique Fernández-Cara, Martin Frank, and Paul Gauché. Field-design optimization with triangular heliostat pods. 21st International Symposium Concentrated Solar Power and Chemical Energy Technologies:

SolarPaces, Cape Town, South Africa, 2015. International Symposium on Concentrating Solar Power and Chemical Energy Systems, SolarPaces.

[11] Shu Geng, Frits W.T. Penning de Vries, and Iwan Supit. A simple method for generating daily rainfall data. *Agricultural and Forest Meteorology*, 36(4):363 – 376, 1986. ISSN 0168-1923. doi: http://dx.doi.org/10.1016/0168-1923(86)90014-6. URL `http://www.sciencedirect.com/science/article/pii/0168192386900146`.

[12] O.D. Jimoh and P. Webster. The optimum order of a Markov chain model for daily rainfall in Nigeria. *Journal of Hydrology*, 185(1–4):45 – 69, 1996. ISSN 0022-1694. doi: http://dx.doi.org/10.1016/S0022-1694(96)03015-6. URL `http://www.sciencedirect.com/science/article/pii/S0022169496030156`.

[13] Harry D. Kambezidis and V. Badescu. MRM: a new solar radiation computing model. Application to Romania. In *VII Conf."Efficiency, comfort, energy preservation and environmental protection", CONSPRESS (publ.)*, pages 195–199, 2000.

[14] Harry D. Kambezidis and Basil E. Psiloglou. The meteorological radiation model (MRM): advancements and applications. In *Modeling solar radiation at the earth's surface*, chapter 14, pages 357–392. Springer, 2008.

[15] Harry D. Kambezidis, B.E. Psiloglou, and C. Gueymard. Measurements and models for total solar irradiance on inclined surface in Athens, Greece. *Solar Energy*, 53(2):177–185, 1994.

[16] Joost-Pieter Katoen and Ivan S. Zapreev. Simulation-Based CTMC Model Checking: An Empirical Evaluation. In *Quantitative Evaluation of Systems (QEST)*, pages 31–40. IEEE Computer Society, 2009. `www.mrmc-tool.org`.

[17] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[18] T. Muneer, M. Gul, Harry D. Kambezidis, and S. Allwinkle. An all-sky solar meteorological radiation model for the UK. In *Proceedings of the Joint CIBSE/ASHRAE Conference*, number 29 in Harrogate, pages 271–280, 1996.

[19] T. Muneer, M. Gul, and Harry D. Kambezedis. Evaluation of an all-sky meteorological radiation model against long-term measured hourly data. *Energy conversion and management*, 39(3):303–317, 1998.

[20] C.J. Noone, M. Torrilhon, and A. Mitsos. Heliostat field optimization: A new computationally efficient model and biomimetic layout. *Solar Energy*, 2011.

[21] Rafael Osuna, Rafael Olavarría, Rafael Morillo, Marcelino Sánchez, Felipe Cantero, Valerio Fernández-Quero, Pedro Robles, Teodoro López del Cerro, Antonio Esteban, Francisco Céron, Juan Talegón, Manuel Romero, Felix Téllez, MaJesus Marcos, Diego Martínez, Antonio Valverde, Rafael Monterreal, Robert Pitz-Paal, George Brakmann, Valeriano Ruiz, Manuel Silva, and Pietro Menna. Ps10, construction of a 11 MW solar thermal tower plant in seville, spain. 13th International Symposium Concentrated Solar Power and Chemical Energy Technologies: SolarPaces, Seville, Spain, 2006. International Symposium on Concentrating Solar Power and Chemical Energy Systems, SolarPaces. ISBN 8478345191.

[22] Basil E. Psiloglou and Harry D. Kambezidis. Performance of the meteorological radiation model during the solar eclipse of 29 March 2006. *Atmospheric Chemistry and Physics*, 7(23):6047–6059, 2007.

[23] Renewable Energy Policy Network for the 21st Century (REN21). Renewables 2015: Global status report, 2015.

[24] C. W. Richardson. Stochastic simulation of daily precipitation, temperature, and solar radiation. *Water Resources Research*, 17(1):182–190, 1981. ISSN 1944-7973. doi: 10.1029/WR017i001p00182. URL `http://dx.doi.org/10.1029/WR017i001p00182`.

[25] R.D Stern and R Coe. The use of rainfall models in agricultural planning. *Agricultural Meteorology*, 26(1):35 – 50, 1982. ISSN 0002-1571. doi: http://dx.doi.org/10.1016/0002-1571(82)90056-5. URL `http://www.sciencedirect.com/science/article/pii/0002157182900565`.

[26] Jingyun Wang, Bruce T. Anderson, and Guido D. Salvucci. Stochastic modeling of daily summertime rainfall over the southwestern United States. Part I: Interannual variability. *Journal of Hydrometeorology*, 7:739–754, 2006.

[27] T. Wendelin. SolTRACE: a new optical modeling tool for concentrating solar optics. ASME, 2003.

[28] Erika Ábrahám, Bernd Becker, Christian Dehnert, Nils Jansen, Joost-Pieter Katoen, and Ralf Wimmer. Counterexample generation for discrete-time Markov models: An introductory survey. In *Formal Methods for Executable Software Models*, pages 65–121. Springer International Publishing, 2014.