

Automatic PV system design using LiDAR data shadow analysis

Delft University of Technology



Automatic PV system design using LiDAR data shadow analysis

by

Joris Bronkhorst

in partial fulfillment of the requirements for the degree of

Master of Science
in Sustainable Energy Technology

at the Delft University of Technology,
to be defended publicly on September 12, 2017 at 01:30 PM.

Supervisor:	Prof. dr. ir. O. Isabella	
Thesis committee:	Prof. dr. ir. O. Isabella,	TU Delft
	Prof. dr. ir. M. Zeman,	TU Delft
	dr. ir. M. Ghaffarian Niasar,	TU Delft
	ir. M. van Hoolwerff,	Solar Monkey

Preface

This document contains the work done for my thesis for the completion of the Master of Science in Sustainable Energy Technology. The work was conducted and completed at the Photovoltaic Materials and Devices (PVMD) group at Delft University of Technology and the company Solar Monkey. During this project I've been working under the daily supervision of Jaap Donkers and Mels van Hoolwerff, both employees at Solar Monkey. The supervisor of the thesis was Dr. Olindo Isabella.

During my study about sustainable energy I was quickly attracted to solar energy. It appealed to me, because it has a wide variety of applications and I found it to be an interesting technology. After completing the solar energy courses from the PVMD group I decided to do my thesis project at that group. I came across the project about the automatic design of PV systems and this appealed to me. It seemed like a project in which new areas had to be explored and in which I could learn a lot. It was also a project in which I could learn some coding, which I hadn't done much during my studies.

A project about automatic PV system design is very large. This provided a lot of room to manoeuvre, but also the difficulty of establishing what the goals would be. There are many aspects that can play a role and many of them would fall outside the scope of the project. Sometimes I had to make sure I didn't dive into something for too long that wouldn't be a part of the project.

I have learned a lot during this project. My knowledge of coding in Python language were absolutely zero when starting this project. Writing thousands of lines of code have really improved my coding skills. Seeing at Solar Monkey how developers work on software also gave me an understanding of the process of software development. I liked the learning process of coding and making algorithms, even though it could also be frustrating at times. Sometimes I felt that the lack of programming experience was dragging down the progress of the project.

The project was mainly about developing new methods. This really forces you to be creative and think about solutions. It was a path in which it was often unclear how the next steps in the development would go. This kept the project very interesting. Development of methods and algorithms does however also have a downside. Trying out ideas could take a long time, especially as an inexperienced coder. If ideas then failed or you realize that it should be done differently it can feel like a slow progress, in which a lot of effort doesn't directly result in something tangible. Looking back at how little I knew at the start of the project and all the working algorithms that have eventually been build up from scratch, it does however feel like useful methods have been created.

There are several people that helped me take a step further in this project. When I didn't really know the next step I asked several people for a meeting to discuss the project and they all agreed to it. First of all I thank dr. David Tax en dr. Marco Loog from the Pattern Recognition Laboratory at TU Delft for meeting with me. Their insight gave me ideas for what to do next. After these meetings I also had a meeting with dr. Karen Aardal from the Applied Mathematics department from TU Delft and with dr. Peter Bosman from the Center for Mathematics and Computer Science in Amsterdam. I also thank those people for taking the time and discussing my project which helped me move forward.

Lastly I thank Jaap, Mels and Olindo for helping me during this project.

*Joris Bronkhorst
Delft, September 2017*

Abstract

The amount of photovoltaic (PV) systems in the world is growing rapidly. The designing process is however still old fashioned and mostly done by hand. Creating a method to automatically design these PV systems can help to stimulate this growing market even further. In this thesis a research is done into the development of an automatic PV system design algorithm.

The research consists of two main parts. A section related to panel placement and a section related to the inverter choice. Strategies were developed for both parts and that resulted in several prototype algorithms. These algorithms were tested to see if these algorithms have practical potential. The panel placement algorithms are divided into two categories: maximum panel placement and finite panel placement. Development of these categories was done for both flat and pitched roof sections. The shading caused by surrounding obstacles was taken into account to find the most optimal positions to place the panels. A grading system for the panel layout was developed to ensure that preferred layouts are found. The shading caused by the surrounding obstacles was also used to determine the type of inverter is that optimal for certain panel configurations.

A proof concept was conducted for the developed algorithms by testing algorithms on real roofs and comparing the results with PV system designs that were made manually. The designs were compared with each other in terms of predicted PV system performance, the layout and the duration of the designing process.

It was demonstrated that the finite panel placement algorithm produced PV systems with an average performance difference of about - 0.83 % with a standard deviation of 5.26 %, compared to the manual designs. The maximum panel placement algorithm took an average of 2.7 minutes with a standard deviation of 2.1 minutes. The finite panel placement algorithm took on average 3.8 minutes with a standard deviation of 5.5 minutes. Both of which can be considered as being significantly faster than making a manual design.

For a PV system with many panels and several string inverters, algorithms were developed that predict the optimal configuration of the strings. It also predicts the total performance losses that occur when panels are connected in series. These algorithms can help to determine the optimal type of inverter and how to optimally configure separate panel strings in large systems. The algorithms work based on initial tests, but not enough testing has been done to be conclusive.

The work done in this thesis can be used as a stepping stone for further development of automatic PV systems design algorithms. The panel placement algorithms and the inverter algorithms can be developed further into a complete automatic PV system design algorithm.

Contents

Preface	iii
Abstract	v
List of Figures	ix
List of Tables	xii
Nomenclature	xiii
1 Introduction	1
1.1 Solar Energy and Photovoltaics	1
1.2 Photovoltaic systems.	2
1.3 Solar Monkey and PVision	3
1.4 Project objectives and thesis outline	4
2 Relevance of automatic PV system design	7
2.1 Overview of the current PV installer market in the Netherlands	7
2.2 Design time savings	7
2.3 Better designs.	8
2.4 Similar tools.	8
3 Design brief	11
3.1 Functionality of the algorithm	11
3.2 Relevant factors in PV system design	12
3.3 Target market.	14
3.4 Algorithm input and output overview.	14
4 Maximum panel placement algorithms	17
4.1 Pitched roofs sections	17
4.1.1 Panel placement without obstacles	17
4.1.2 Panel placement including obstacles.	23
4.2 Flat roofs.	26
4.2.1 Optimal tilt angle.	26
4.2.2 Self-shading and row distance	26
4.2.3 Orientation of the panels	27
4.2.4 Maximum panel placement including obstacles.	30
5 Finite panel placement	33
5.1 Introducing shadows.	33
5.1.1 The PV performance grid	33
5.1.2 The performance of a PV modules	34
5.1.3 The performance of a PV system.	37
5.2 Brute force methods	38
5.3 Genetic algorithm	41
5.4 Local optima search and combine algorithm.	42
6 PV system layout grading	49
6.1 Performance aspects.	49
6.2 Installment aspects	49
6.2.1 Number of panel sections	49
6.2.2 Vertical or horizontal placement.	50
6.2.3 Compactness of a panel section	50
6.2.4 Mounting material	51

6.3	Aesthetic aspects	52
6.4	The grading system	53
7	Inverter optimization options	55
7.1	Inverters	55
7.1.1	Central and string inverters	55
7.1.2	Micro inverters	56
7.1.3	Power optimizers	57
7.2	Panel miss match.	57
7.2.1	Relative intensity, obstacle view, and sun tracks	58
7.2.2	Miss match method 1	60
7.2.3	Miss match method 2	64
7.3	String building in large PV systems	65
7.4	Optimal string inverter sizing	70
8	Proof of Concept	73
8.1	Methods of comparison	73
8.2	Extract roof sections and obstacles	75
8.3	Performance grid and shape corrections	78
8.4	Algorithm settings	81
8.4.1	Finite panel placement algorithm	81
8.4.2	Maximum panel placement algorithms	82
9	Proof of concept results	83
9.1	Finite panel placement algorithm for pitched and flat roofs	83
9.1.1	Finding the manual design and/or alternatives	83
9.1.2	Differences in the performance between the designs	87
9.1.3	Algorithm duration compared to manual designing	88
9.2	Maximal panel placement algorithm for pitched and flat roofs	91
9.2.1	Azimuth differences algorithm and manual designs	91
9.2.2	Differences in the amount of panels.	92
9.2.3	Algorithm duration compared to manual designing	94
9.3	Finite panel placement algorithm for a panel range	96
9.3.1	Diversity among the produced solutions	96
9.3.2	Algorithm duration compared to manual designing	97
9.4	Results summary.	97
10	Conclusions and recommendations	99
10.1	Conclusions	99
10.1.1	Panel placement algorithms	99
10.1.2	Inverter algorithms.	100
10.2	Recommendations	101
10.2.1	Current algorithm improvement and optimization	101
10.2.2	Future steps for automatic PV system design	104
10.2.3	Large scale implementation.	105
10.2.4	Small scale implementation	106
	Bibliography	109
	Appendix A - Genetic Algorithm	111
	Appendix B - Pseudo code of Algorithms	115
	Appendix C - Inverter sizing table	119

List of Figures

1.1	Total globally installed capacity of PV installations.	2
1.2	Schematic of the components in a residential grid-connected PV system.	3
3.1	Overview of some of the roof shapes on buildings that can be encountered.	12
3.2	A flowchart of the inputs and the output of the algorithm that will be developed.	15
4.1	The result of the recursive partitioning algorithm placing a maximum amount of panels on a rectangle roof section	18
4.2	Maximum panel placement in landscape orientation on a rectangle roof section.	19
4.3	Maximum panel placement in portrait orientation on a rectangle roof section.	19
4.4	Maximum panel placement with multiple orientations on a rectangle roof section.	20
4.5	Maximum panel placement with a single orientation on a triangle roof section.	21
4.6	Maximum panel placement with multiple orientations on a triangle roof section.	21
4.7	Maximum panel placement with single or multiple orientations on a trapezium roof section.	22
4.8	Placement of a panel grid on a non standard roof surface. Landscape orientation on the left and portrait orientation on the right.	22
4.9	Panels that fit within the roof polygon are colored displayed in green.	23
4.10	Panels that don't fit within the roof polygon are removed.	23
4.11	A rectangle roof with obstacles divided into sections without obstacles.	24
4.12	Maximum panel placement for a rectangle roof with obstacles.	24
4.13	Maximum panel placement for a rectangle roof with obstacles with extra shifts to place more panels	25
4.14	Placement of extra panels that have different orientation than nearby panels.	25
4.15	Maximum panel placement for a non standard roof including obstacles.	26
4.16	The shadow caused by a panel on flat roof.	27
4.17	A south orientation is changed to an east-west orientation, because of large obstacles in the south.	28
4.18	A south orientation is adjusted to a south-south-east orientation due to large obstacles in the west.	28
4.19	The obstacle view for a roof near the EEMCS building.	29
4.20	The normalized performances of a panel with a tilt angle of 12 degrees and various azimuths, with and without the surrounding obstacles included in the algorithm.	29
4.21	Maximum panel placement for a flat roof with obstacles for both landscape and portrait orientation with the panels facing south.	30
4.22	The angle α that defines the roof orientation.	31
4.23	Maximum panel placement for a flat roof with obstacles with panels orientated in line with the roof in two different ways.	31
4.24	Maximum panel placement for a flat roof with obstacles with panels back to back in three different versions.	32
5.1	An artificial rectangle roof surface with two obstacles and three shaded areas.	33
5.2	An artificial performance grid for the roof of figure 5.1.	34
5.3	The IV and PV curve of a PV module.	34
5.4	The effect of irradiance and temperature on the IV curve of a PV module.	35
5.5	An example of a solar panel with the cells visualized.	36
5.6	Two panels on top of the performance grid, with the red dot as the center point of the panel.	36
5.7	Dividing the panels into three horizontal sections.	37
5.8	The effect of series and parallel connections on the total IV curves.	38

5.9	A clustered, but disconnected group of panels.	39
5.10	Examples of non rectangular panel sections.	39
5.11	Illustration of shifting a 2x3 panel section across a roof surface.	40
5.12	The amount of panel section combinations, up to two sections.	40
5.13	A flowchart of the stages and the steps of the local optima search and combine algorithm.	42
5.14	Points saved after a probing algorithm is applied. Green areas receive the least amount of shading	43
5.15	Panel sections to be placed and roof positions.	44
5.16	Illustration of how duplicate solutions can arise.	45
5.17	Adding an extra edge to the panels will ensure the row distance.	46
5.18	The extra edge on panels can cause a false positive when checking for obstacle overlap.	46
6.1	Layouts with different number of sections for a PV system with eight panels.	50
6.2	Horizontal versus vertical placed panel sections.	50
6.3	Two different panel sections with 10 panels with different compactness.	51
6.4	The convex hull for three different panel layouts.	51
6.5	Mounting rails behind the panels.	52
6.6	Solutions placed in a graph with the factors L and S on the axes.	53
7.1	A string or central inverter PV system.	56
7.2	A micro inverter PV system.	56
7.3	A power optimizers inverter PV system.	57
7.4	Differently moving shadows, but with the same total shading per panel.	58
7.5	The relative intensity for a module oriented south with a tilt angle of 35 degrees.	59
7.6	The obstacle view at a certain location.	59
7.7	The sun tracks for the months January and May at a certain location.	60
7.8	Two different artificial obstacle views.	60
7.9	Two different artificial obstacle views.	61
7.10	Two artificial obstacle views for panels facing south.	62
7.12	A PV system on a pitched roof with the obstacles views from three panels.	63
7.13	A PV system on a pitched roof with a dormer.	63
7.14	Calculation of the two different performances for miss match method 2	64
7.15	Miss match of a panel versus another panel with a different orientation.	65
7.16	A panel section of 16 panels that is used to test the string building algorithm.	66
7.17	A panel selection is transformed to a cascaded union.	67
7.18	Optimal string configurations for two and three separate strings for a system with 16 panels.	67
7.19	A panel section of 126 panels divided into 7 parts and then in into 21 strings.	68
7.20	A panel section represented as a graph, with strings as paths.	69
7.21	Weighted graph with strings as paths and miss matches as weights between connections.	69
7.22	DC power and energy distribution or a PV system facing south.	70
7.23	DC power and energy distribution or a PV system facing north.	70
7.24	The inverter sizing ranges for different installation characteristics.	71
8.1	Blocks of AHN data that can be downloaded and a sample of PV system locations.	75
8.2	The AHN layer in QGIS with the red dots representing roofs with PV systems designed with the Solar Monkey software.	76
8.3	A PV system designed with the Solar Monkey software and the corresponding AHN layer.	76
8.4	Drawing the roof polygon and dormer in QGIS on the AHN layer.	77
8.5	A very large flat roof, with corresponding AHN layer and drawn polygons.	78
8.6	The WGS coordinate system and the RD coordinate system.	78
8.7	A grid drawn with python and the coordinates shown on the AHN layer in QGIS.	79
8.8	Difference between the drawn polygon and the actual roof.	79
8.9	Roof azimuth and pitch corrections.	80
8.10	The calculated performance grid with interpolation of the missing values.	80

9.1	Percentages of how often the algorithm found the manual design, with and without panel size adjustments.	84
9.2	An example of a roof without obstacles that requires a non-standard solution	84
9.3	A panel section doesn't find a place, because the starting positions are not quite in the right spot.	85
9.4	Different solutions for a roof with 12 panels to place.	85
9.5	Three solutions for a flat roof with 12 panels.	86
9.6	Some solutions to a flat roof when placing a limited amount of panels with a south orientation.	86
9.7	Some solutions to a flat roof when placing a finite amount of panels, in an east-west orientation.	87
9.8	The performance differences with the manual designs for the pitched roofs with a finite amount of panels.	87
9.9	An example of very similar designs, but with a performance difference of almost 14%.	88
9.10	The correction of a manual design, due to the bad placement on the AHN layer.	88
9.11	The computation times of the pitched roof with a finite amount of panels.	89
9.12	The duration of the algorithm when the amount of panels is increased.	90
9.13	Two different roof scenario's that prompt extra panel sections due to an obstacle.	91
9.14	The absolute differences in the orientation determined by the algorithm and the one chosen in the manual design.	92
9.15	A manual and automatic design for a flat roof PV system.	92
9.16	Two manual designs on pitched roofs and a maximum panel placement algorithm result.	93
9.17	The total amount of panels that fit for different panel azimuths.	94
9.18	The computation times for flat roofs with different surface areas.	95
9.19	Three flat roof surfaces and their computation time.	95
9.20	Different solutions for a panel range of 8, 9, 10.	96
10.1	Example of a roof where extra shifts result in extra panels.	102
10.2	The AHN layer and a aerial photograph could be combined to draw the roof polygons.	106
5	Optimum inverter sizing % for different orientations and tilt angles.	119

List of Tables

9.1	Computation times with different step sizes.	90
9.2	Calculation times for different design preferences.	97
9.3	Solution finding rates for the finite panel placement algorithm for 41 pitched roofs. . .	97
9.4	Results of the finite panel placement algorithm for 41 pitched roofs.	97
9.5	Results from the maximum panel placement algorithm for 15 flat roofs.	98

Nomenclature

Symbol	Description	Dimensions	Units
α	Angle with north-south axis	–	°
β	Tilt angle of a panel	–	°
d	Row distance	L	cm
H	Hull factor	–	–
h	Distance covered by tilted panel	L	cm
I	Current	I	A
L	Rail length	L	cm
P	Performance	ML^2T^{-2}	kWh
S	Number of sections	–	–
V	Voltage	$ML^2T^{-3}I^{-1}$	V
w	Width of a panel	L	cm
W_p	Watt peak	ML^2T^{-3}	W

1

Introduction

Energy consumption is the big driving force in our day to day life. Everything that we do requires energy. It heats our houses on a cold day and powers the air conditioning during a hot day. Energy is required for transportation, the production of food, industry, lighting and much more. Total global energy consumption in 2015 was about $150 \cdot 10^{12}$ kWh [1]. This energy consumption is expected to increase to about $240 \cdot 10^{12}$ kWh in the year 2040 [2]. The fastest growing source of energy in 2015 was renewable energy with a growth of over 15% [1].

The world is gradually moving towards a more sustainable energy economy. There are several factors that drive this transition. In recent decades the climate change debate has become more dominant. With more and more evidence supporting the claim that the increased CO_2 in the atmosphere due to fossil fuels will cause a climate change. In 2016 the antarctic hit a CO_2 concentration of 400 parts per million (PPM), which is the first time in four million years that it has reached that level [3]. This high concentration of CO_2 will very likely cause an increase in global temperature, which will have disastrous consequences for certain parts of the world. Governments are becoming more aware of this danger and are trying to limit their emission of CO_2 . This is supported by the recent Paris agreement on climate change of December 2015. The countries responsible for 97% of the global emissions have made significant commitments to prevent this climate change, which includes reducing their carbon emissions [4].

The amount of global investment in renewable energy is rising each year and a record breaking \$285.9 billion was invested in renewable energy in 2015 [5]. For the first time a majority of all the installed energy capacity in a year was renewable, namely 53.6% [5].

1.1. Solar Energy and Photovoltaics

Many technologies are being developed that can generate renewable energy. Solar energy is source of renewable energy that can be harvested in multiple ways. Several examples are that the sun can be used for heating green houses, for water heating or the sunlight can be used with photovoltaic (PV) modules to create electricity. Photovoltaic modules are also named solar panels, or sometimes just panels for short. Solar panels can generate energy by converting sunlight into electricity. It is one of the most promising renewable energy technologies around today [6]. The production costs of PV modules have decreased a lot over the years, due to advancement in production technologies.

There are many benefits to generating energy with solar panels. The energy is coming from the sun, which is free and available for everyone and everywhere. The solar panels are not harmful for the environment, they are reliable, safe and are easy to install. Another big benefit is that solar panels can be used on a huge scale, but also on a very small scale. This makes the technology attractive for a wide range of applications.

The trend of cheaper solar panels makes the installation of new PV systems more popular. The International Energy Agency (IEA) does research into the growth of renewable energy. The globally installed PV capacity in 2015 was at least 227.1 GW [7]. The IEA has a program, called the Photovoltaic Power Systems Program (PVPS), with 29 countries in which it tries to enhance the role that PV energy

plays in the transition towards renewable energy. The amount of globally installed capacity is tracked each year and an impressive growth can be seen since the start of the century. This increase in capacity can be seen in figure 1.1.

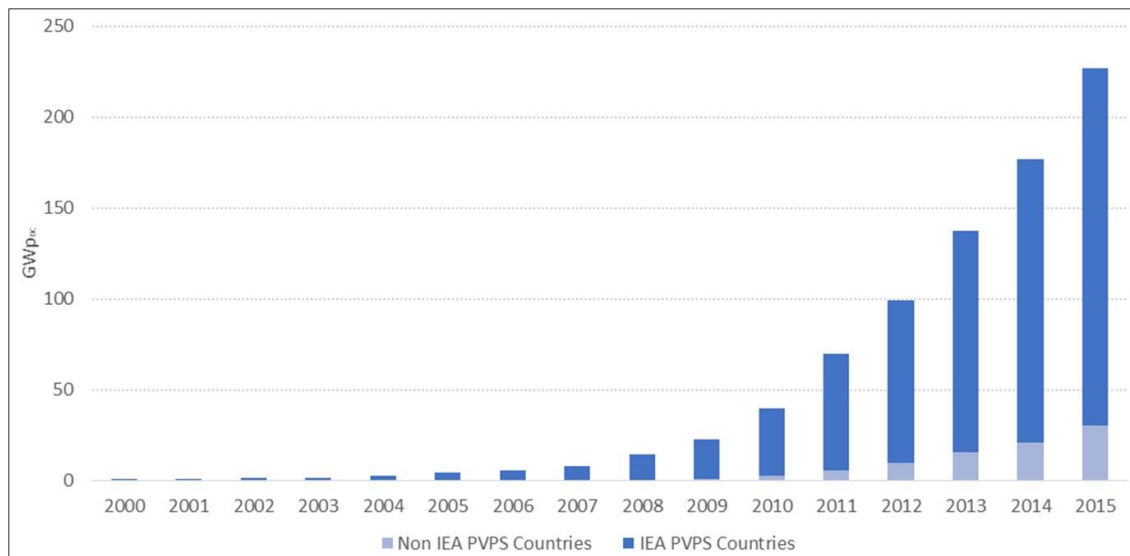


Figure 1.1: Total globally installed capacity of PV installations. Figure taken from [7].

This increase in installed capacity each year indicates that the PV system market is blooming. This makes looking into the design of PV systems very interesting. With all of these new systems installed each year, the method used to design these systems is important.

1.2. Photovoltaic systems

A PV system contains all of the components required to generate electricity from sunlight with the use of PV modules. Not only the PV modules are required, but also several other components, which combined are called the Balance of System (BoS). It contains the mounting structure, an inverter and cables. The inverter will change the Direct Current (DC) output of the PV modules into Alternating Current (AC) output that can be fed into the grid or can be used by home appliances. It is an important component of the PV system and comes in many different types and often also has other functions. The inverter is discussed in more detail in chapter 7.

PV systems can be divided into three main categories: grid-connected systems, stand-alone systems and hybrid systems [8]. A grid connected PV system is connected to the electricity grid. This means that the electricity produced by the PV modules is fed into the electricity grid after it is converted to AC. A stand-alone system is not connected to the electricity grid, which is why it can also be called an off-grid system. These systems will almost always have a battery system, since no power can be generated at night. This means that energy storage is needed to have electricity available at all times. These systems will have a battery system and a charge controller. A battery will get damaged or its lifetime will be reduced if it is not (dis)charged in a proper way. This can be due to overcharging the battery, charging too fast or by discharging the battery too much. The charge controller will make sure that the (dis)charging of the battery will be done without damaging the batteries. A hybrid system has an additional generator to generate electricity besides the PV modules. This can be wind energy or a diesel engine.

In this thesis the focus lies on PV systems that are grid-connected and are placed on building roofs. In such a PV system a certain part of the generated power is used for home appliances and another part is fed into the electricity grid. For such PV systems the AC power from the inverter is fed to a distribution board which feeds it to either the grid or to home appliances. A schematic of such a PV system can be seen in figure 1.2.

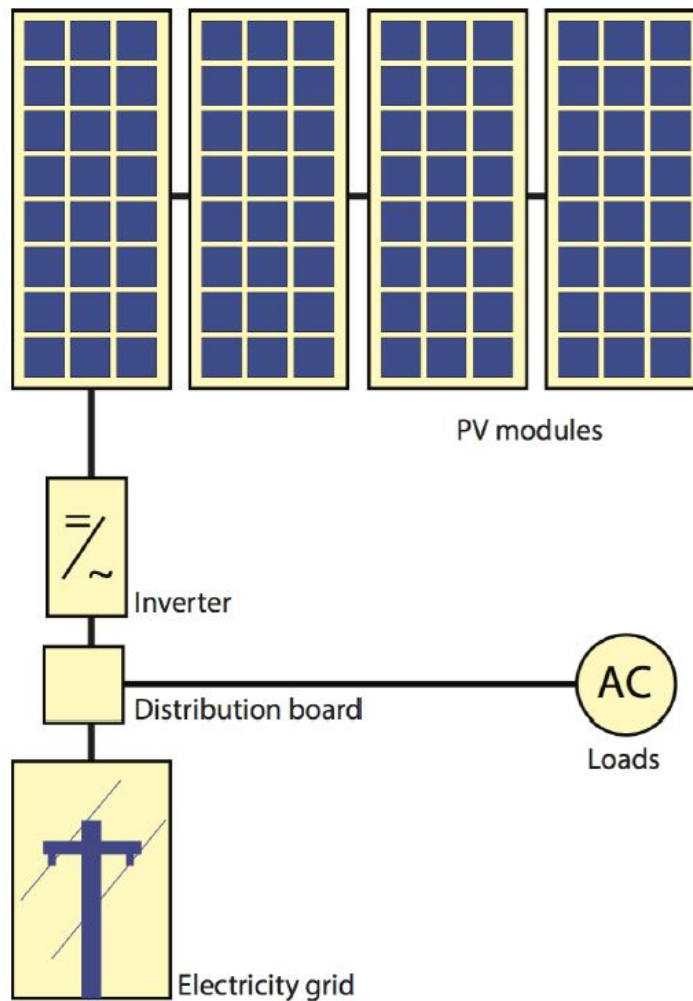


Figure 1.2: Schematic of the components in a residential grid-connected PV system. Figure taken from [8].

1.3. Solar Monkey and PVision

Solar Monkey is a company that specializes in advanced software that can design PV systems and predict their performance. The company sells licenses to use the software to PV installers. They also have a foundation called Zonnegarant that can give the owner of a PV system, developed with their software, a guarantee on their energy production. This guarantee is a service that is made possible by accurate energy yield predictions by their software. One of their services is also a continued surveillance of the PV systems that were designed with their software. This ensures that a possible defect is noticed right away and can be fixed fast. They continue to improve the software to make the designing process easier, more optimal and to make better predictions for the energy production.

Performance calculations based on LiDAR data

The software from Solar Monkey uses aerial photographs and LiDAR (Light Detection And Ranging) 3D data to design the PV systems. This LiDAR data is a height map of the Netherlands with a resolution of half a meter. The solar panels can be drawn directly on a photo of the roof. After a design has been made the software can predict the performance, based on the position, orientation, tilt angle and it performs a shadow analysis with the surrounding obstacles. The surrounding obstacles are determined by analyzing the height data. The software will make a recommendation for possible inverters that are compatible with the designed PV system. This design method is currently done manually by dragging and dropping panels on the roof.

PVision

Solar Monkey has the vision to have software that can perform this designing process automatically. The software can then make a suggestion for a roof based on design preferences. In order to do that, an automatic PV system design algorithm has to be developed. This software will automatically design a PV system for a rooftop and calculate the output.

This development is part of a project called PVision, which is a collaboration between several partners. These partner are Solar Monkey, Readaar, TU Delft and Universiteit Utrecht (UU). Each of these partners is working on an certain aspect that is required for the total result. Solar Monkey works on the development of software to automatically design and calculate the best performing PV system on a given location with certain preferences as input. Readaar is a company that works on the development of software that can analyze aerial photographs and 3D LiDAR-maps. This is required to obtain information about the roof on which the solar panels have to be placed. These analyses also contribute to the precision of the output predictions performed by Solar Monkey. The TU Delft performs scientific research into the recognition of albedo from aerial images, which has an impact on the performance. The TU Delft as well as the UU perform scientific research in the modelling of the energy yield of PV systems and they monitor existing PV systems and analyze the performance data.

1.4. Project objectives and thesis outline

This thesis is the automatic PV system design aspect of the PVision project. It is the start of the process of the developing an algorithm that will automatically design PV systems. The research goal of this thesis is:

Perform research into the development of an automatic PV system design software and create a prototype in python language.

This document was written to show the work that was done in order to reach the project objective. Ideas and concepts are described and several tests are discussed and evaluated. This report showcases all the work that went into the development process. It discusses and evaluates the results and explains the reasoning behind decisions that were made in subsequent parts of the research.

Some of the panel placement algorithms, the proof of concept algorithms as well as the algorithms concerning inverters can be found in pseudo code in Appendix B. The actual code was not included, as it would make for quite a large addition to the document and it is still prototype code that will have to be rewritten for the most part.

Development orientation

The first two chapters lay the groundwork for starting the development progress. The relevance of developing an automated PV system design algorithm is discussed in chapter 2, this is done by looking into the current PV installer market of the Netherlands and the benefits such an algorithm could have. A design brief for the proposed algorithm is discussed in chapter 3, which is about the functionality of the algorithm and the factors that play a role in its development.

Development of the algorithms

The development of the algorithms can be divided into two parts, a part about the placement of panels and a part about the choice for the inverter. The panel placement part is most of the report and starts out with very basic artificial roof scenario's. More complexity is added throughout the project as it evolves into an algorithm that is eventually evaluated with real roof scenario's. The development starts with the simplest case of filling a roof with the maximum amount of panels that will fit in the first section of chapter 4. A roof should however not always be filled with a maximum amount of panels. It is often the case that just a finite amount of panels is desired. The algorithms that were developed for placing a finite amount of panels are discussed in chapter 5. This starts out by introducing the importance of shading on the roof and how this is included in the algorithm. Since the roof isn't maximally filled, the position and layout of the system gain a lot of room for variation and optimization. All of the strategies

were first made for pitched roofs and were later adapted to be applicable on flat roofs. In finding the most optimal PV system for a certain roof, the layout is very important. There are several aspects that play role in the layout. These were examined and their relevance was considered in finding solutions. These considerations are bundled together in chapter 6. In chapter 7 some basic theory about inverters is discussed and from that several theoretical methods are developed that can determine a choice for the inverter.

Proof of Concept

The panel placement algorithms were tested and evaluated on several aspects. This evaluation was done by running the algorithm on real roofs that also had a manual design for it. A comparison between the algorithm output and the manual designs was made in terms of PV system performance, layout and design time duration. The methodology for running the algorithm on a real roof, as well as the method to evaluate the performance of the algorithm can be found in chapter 8. The results of these tests are discussed in chapter 9.

Conclusions and recommendations

The conclusions from the proof of concept are drawn in the last part of the report. Several different aspects were evaluated and the results are discussed in chapter 10. This project presents a prototype and many recommendations for improvements and next steps to be taken are presented.

2

Relevance of automatic PV system design

In this chapter the relevance of an automatic PV system design algorithm with respect to the PV installer market is discussed. First an overview of the current PV system installer market in the Netherlands is given. An entire separate research could be done about this, as there are many factors that play a role in it. This chapter is however only meant to gain an understanding of how this market looks like. After this brief introduction to the market, two potential benefits that come with automated PV designs are discussed. These aspects are some of the driving forces behind the development. These factors are the time savings and the creation of potentially better PV systems than can be designed manually.

2.1. Overview of the current PV installer market in the Netherlands

There are many installers of PV systems currently active in the PV market. In the Netherlands about 1000 to 1500 companies are installing PV systems [9]. These installers all have their method to design a PV system for their costumers. The information about the companies in the current market was obtained by contacting some companies themselves, or from the information on their website. The information is not very specific, as the goal was just to get an idea of what the current market looks like. The installing companies in the current market differ significantly in size. Some companies are run by just one man, while other companies can have over 1000 employees. These very large companies are usually not just specialized in installing PV systems, but also do other things as well. The companies that are active in the current market also differ a lot in the amount of PV systems that they install. There are small companies that only install about 50 PV systems each year, but there are also large companies that install over 1500 of them each year. These companies can also have very different methods of operation. Some companies use their own advanced design and calculation software or have a license to use software from a company specialized in PV system design software, such as Solar Monkey. Some companies don't even use software at all and use very simple techniques to design their PV systems, which includes rules of thumb and making measurements on the roof by hand. The working strategy of the companies in the current market is therefore by no means fixed, which means new approaches could be adopted. One of those new approached would be to have the design of PV systems done automatically or to use suggested designs from software. An automatic designing approach could increase activity in the entire market, as it will be easy to use and more designs can be created in a shorter time span.

2.2. Design time savings

There are various factors that have to be taken into account when designing a PV system. These factors are physical, environmental and economical requirements and constraints, as well as the properties of

all the components in the PV system. Due to all of these factors the PV systems are custom made for a specific situation. The designing process can be time consuming, especially if the installers do not use software for this.

In the current market with a lot of installing companies, the competition is fierce. Many people that are interested in solar PV will contact several installers and ask for a possible design. A result of this is that companies will have to make designs for potential costumers that in the end don't order anything. Companies offering free designs will then spend time on something that doesn't earn them money. A rough estimate is that only about 15% to 25% of all the designs made by companies are actually build.

Some installers, especially smaller ones, will visit the site to examine the roof and the surrounding obstacles. This is a time consuming process as it will take several hours. Several design plans are made and then eventually one of them is chosen. The average time spend on a design can vary quite a lot between the companies. A common method do determine the size of the roof for example is to measure and the count the tiles on the roof. Software that some companies use can estimate the size with aerial photos much faster. The time ranges between 10 minutes and 45 minutes, with the smaller companies usually taking more time. In a bad case scenario this would mean that 6 out of 7 designs that all took 45 minutes to make are not actually build, this is a time of 4.5 hours of work.

If the system is designed automatically, a lot of time can be saved. The design will not only be made faster, but it no longer requires human attention and you can do something else while the algorithm is running. Time savings become more significant at a large scale. It may be become possible to design PV systems for entire streets. It can then present designs for all roofs within a reasonable time frame and without human effort.

2.3. Better designs

An automated design algorithm that takes shading on the roof into account could perhaps find better solutions than a human would. The algorithm could calculate the exact optimal position for the panels on the roof. This minimizes the amount of shading that the panels receive and results in a higher performance. Not only the panel positions can be better, the choice for the inverter might also be better calculated. Even a slightly better overall performance of the PV systems can have large impact on a large scale. If all the installations are only one percent better it can already be a boost to the total generated solar energy. If all the dimensions of the roof are correct it could find panel layouts that a human might not have thought of. This could present preferable layouts that would otherwise have never been found. These possible gains are an incentive to investigate automated design algorithms.

2.4. Similar tools

Because solar power is growing, there are a lot of parties developing different kinds of software tools related to solar PV. Most of these are about estimating the PV potential for a certain roof. There are a lot of online applications and software packages. These software tools vary greatly in complexity. Most of them are relatively simple approximations of the PV potential on a certain location based on the location, tilt and orientation. These tools don't take into account any shading losses from surrounding obstacles. These tools only provide an estimation of the possible performance, but don't make a panel layout. This means only a performance number is provided and sometimes also a financial picture is sketched. It cannot be used to determine where the actual panels on the roof should go. Some tools like the System Advisor Model (SAM) from the National Renewable Energy Laboratory (NREL) are very complex and require the user to create a 3D model of surrounding obstacles to perform shading analysis. This software can be accurate in the performance prediction, but designing the systems is not automatic and very time consuming.

Google Project Sunroof

One of the software tools to estimate the PV potential for a roof is Google's Project Sunroof, released in 2015. This tool uses Google's aerial images and combines a 3D model of the house and surrounding obstacles with weather data to analyze the PV potential for a roof [10]. It is a web application that takes an address in the United States and calculates the PV potential and financial aspects. It does

however not offer a layout of panels that fits on the roof. This means a complete layout cannot be automatically generated. This is one of the core aspects of the algorithm developed in this thesis. This is where the algorithms developed in this thesis can differentiate itself from all these other tools.

3

Design brief

In this chapter a design brief for the automated design algorithm is presented. The first part of the design brief consists of what the desired functionality of the algorithm is. This section is about the technical development goals that are set for the final algorithm. The second part is about the relevant factors that play a role in PV system design. Not all of these factors are within the scope of this thesis however. The third part is about who the main users of the algorithm could be and the actors that could benefit from it. In the last part an overview of the input and output of the algorithm is given.

3.1. Functionality of the algorithm

Very broadly spoken the algorithm should present possible PV systems for a given roof that meet the PV system requirements. In order to do that the algorithm has to receive and process information about the roof to make sure that the physical constraints are well established. These physical constraints as well as the types of roofs the algorithm should be able to handle will be discussed in more detail in second section of the design brief. This section is about the non physical constraints, namely the design preferences.

The design preferences of the PV system determine the way that it has to be designed. There can be several different design preferences for a PV system. Four of them are treated in this thesis and should eventually be incorporated in the design algorithm. These preferences are:

1. Maximize the amount of panels on the roof.
2. Stay within a certain budget.
3. Get a certain yearly energy yield.
4. Reach a certain return on investment (ROI).

These preferences can be divided into two main categories: maximum panel placement and finite panel placement. The first preference is a very trivial scenario in which simply the most amount of panels that fit is placed on the roof. Preferences two and three will result in a certain range of panels that is to be used. It could for example result in a design preference of using 8 to 10 panels. Beforehand it is unknown how many panels are needed to reach a certain energy yield, which means an estimation is done and then a range of amounts of panels is used near that estimate. When a budget is considered it might not always be the best to use the maximum amount of panels within the budget, as that might not work out well for that particular roof. It could result in very awkward layouts or panels in positions that receive a lot of shading. This will therefore also result in using a certain range of possible amounts of panels. The last preference is about making an investment and expect a certain ROI. This preference is a variation on the first preference if no budget limit is given. Then the panels performing below a certain threshold can be removed to ensure a certain ROI is reached. If there is also a limit on the budget, it will become more similar to the second objective.

The aesthetics are another important constraint for the algorithm. The best performing system will often not be the most pretty. There is a trade off between aesthetics and performance. In a final version of the algorithm there should be an option to chose how much aesthetic factors are be taken into account.

A benefit of automatically designing the systems is to have the algorithm design for multiple houses, or entire streets. This will require the algorithm to take in all the relevant information about the roofs and design preferences. It should then also automatically chose the most optimal roof surface for each address. It should give an overview of the performances of all the addresses. This is however an aspect that is beyond the scope of this project. This thesis is about starting the development of the first algorithms and this aspect will come later.

3.2. Relevant factors in PV system design

There are many factors that have an effect on the performance of a PV system and those could all be considered in the designing process. Not all of these factors are however relevant enough that they will influence the designing process. There are also several practical and physical constraints that puts limitations to the design. In this section the factors that can play a role are briefly discussed and their relevance is determined. Some aspects are only mentioned, as they fall outside the scope of this project, but could however be investigated more in the future to create a more accurate algorithm. In order to give this section more structure, the aspects are divided into several categories. These categories are physical constraints, performance aspects, legal and financial aspects and aesthetic aspects.

Physical constraints

The most trivial constraint is the roof itself. Roofs come in many shapes and sizes and can have physical obstacles, such as windows or dormers. The algorithm should recognize the shape of the roof and identify the physical obstacles. How these are identified is not a part of this thesis, but is part of the research that is performed by Readaar. The starting point in this thesis is that these roof shapes and obstacles are given. The algorithm should make sure that panels do not overlap with windows or dormers and that everything fits on the available roof area. The algorithm should also be able to work on most roof types. A few broad categories are defined in order to get a structured overview. Some of the possible roofs that can be encountered can be seen in the figure 3.1.

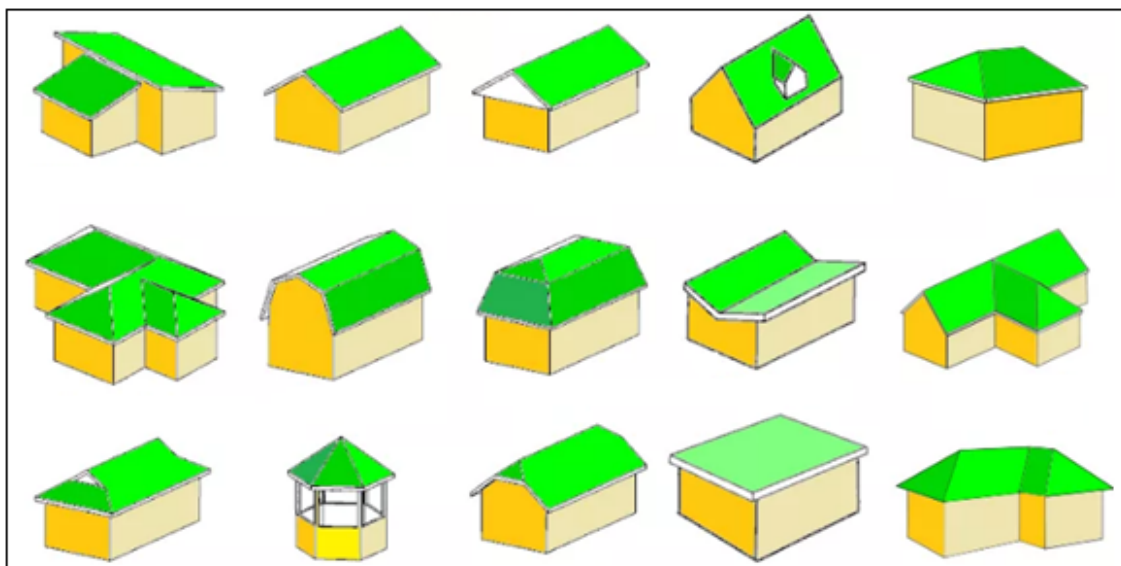


Figure 3.1: Overview of some of the roof shapes on buildings that can be encountered. Figure adapted from [11]

Panel placement does however not work on the entire roof, but on a roof section. A closer look to

figure 3.1 shows that the variety of shapes of the roof sections are quite limited. In this thesis the roof sections that are considered are either flat or pitched. The shapes of the roof sections are divided into four categories namely: rectangle, triangle, trapezium and non-standard. The non-standard roof section has a shape that does not belong to any of the other three categories.

A big consideration for the roofs is that they are often not free of obstacles that can block placement of a panel. The roofs can also be distinguished from each other by looking at whether they have physical obstacles such as windows, dormers or chimneys.

Performance aspects

There are several different aspects that have an effect on the performance of a PV system. The most prominent factor is the amount of solar irradiance on the panels and how often the panels are shaded. The temperature also plays a role and both contributions are mentioned in chapter 5. There however many more factors that have an influence on the performance, as is discussed in for example the paper by Ekici and Kopru [12].

In this paper they focus on cable losses, but also examine all other system losses that can occur. Not all of these losses can be prevented with the design of the system, but can be taken into account with a prediction for the yield. These factors are thermal losses and unavoidable shading losses due to clouds. Another factor is the collection of dust and dirt on a panel, which can reduce the performance by up to 15% in extreme cases with little rainfall [12]. This is however more a good incentive to clean the solar panels once in a while. There are however also some losses that could be minimized in the designing process and these are cable losses, inverter losses and module mismatch losses. The cable losses are related to the length of all the cables and therefore making a layout that minimizes the required amount of cables will result in fewer losses. It should however be noted that the total cable losses for a standard residential PV system is only about a few percent [12] and that small reductions of cable length will have only a minimal effect. The inverter losses could be minimized by choosing the most optimal inverter for the designed system. The module mismatch losses occur when panels are connected in series and do not produce the same amount of energy at the same time. This is not always avoidable, but a clever watch out for partial shading due to surrounding obstacles could minimize this.

Legal and financial aspects

Financial aspects are mostly the constraint of having a certain budget to design a PV system. It could however also be that case that a certain financial return on investment (ROI) is desired. In that case an analysis has to be done that determines the yield a panel of a certain type needs to have in order to achieve that ROI. The legal aspects that play a role are the net metering or subsidies and that can be different for each country. For the most complete algorithm it should be aware of the regulations at the location of the system. This is however not in the scope of this thesis and could be an entire separate research. A note about the Netherlands is that the net metering scheme has recently been extended from 2020 to 2023 [13].

Aesthetic aspects

One of the barriers for people when it comes to placing solar panels on their roof is the way that it will make their house look. The choice is mostly between just black or blue panels and people fear the appearance of their house will be disturbed [14]. The conference paper by Ad Breukel discusses the results of a survey that showed that aesthetic aspects play an important role for people when choosing for solar energy. It is contradictory to the idea that people just go for the cheapest and best price option, they tend to be very interested in an aesthetically appealing solution. In the first stages of the algorithm development, there isn't room for very aesthetically special options, such as building integrated PV or special panel types. Only simple rectangle panels will be used, but attention has been given to how the layout of the panels looks like.

3.3. Target market

In the development of a product, which the algorithm in essence is, it is important to think about the people that could benefit from it. This section discusses the actors for which the algorithm can be beneficial.

Private individuals

Installing PV modules on your own house is becoming much more popular over the past years. More and more people are deciding to place solar panels on their roof. The completed algorithm can give people a fast suggestion for a PV system for their roof within their personal budget. Having such an easy and fast option to get a suggestion can make more people look into the options for solar panels on their house. If more people look into this possibility, it is likely that more people will end up having a PV system installed.

Housing corporations and real estate owners

Solar panels can be seen as a good investment, because in the long run the invested money will be earned back with the generated energy. Installing solar panels is attractive to those that own a lot of real estate. An example of this are housing corporations. According to a research published in April 2016, the amount of homes owned by housing corporations in the Netherlands is 2.2 million [15]. This means that they will also have a large amount of roofing area available to them. This means that they could invest in a lot of solar panels on those roofs. Other than the housing corporations are simply other real estate owners that have a lot of property. An algorithm that can automatically design PV systems for many of their buildings can provide interesting investment opportunities.

Governments

Governments are continuously driven towards more renewable energy goals. All the more reason to look at all possibilities in that regard, including solar panels. A report from Holland Solar states that about 21 square kilometers of useful roofing surface for solar panels is still available on government buildings [16]. This means that a lot of roofing area owned by the government could be used for solar panels. If the software can provide designs for this roofing area, then that could be of interest for the Dutch government. This can present the government with an option of the implementation for renewable energy on a large scale, which can help with achieving the energy goals. Since solar panels are also an investment, it will also present the government an interesting and responsible investment opportunity.

3.4. Algorithm input and output overview

When developing an algorithm it is important to consider the input and the output of the algorithm. There are two categories of inputs for this algorithm. The first one is a location specific input that includes the roof characteristics and the environment characteristics. The second category is the design preference for the PV system. These inputs will ultimately determine the algorithm parameter settings. The overview of the inputs and the output can be seen in figure 3.2.

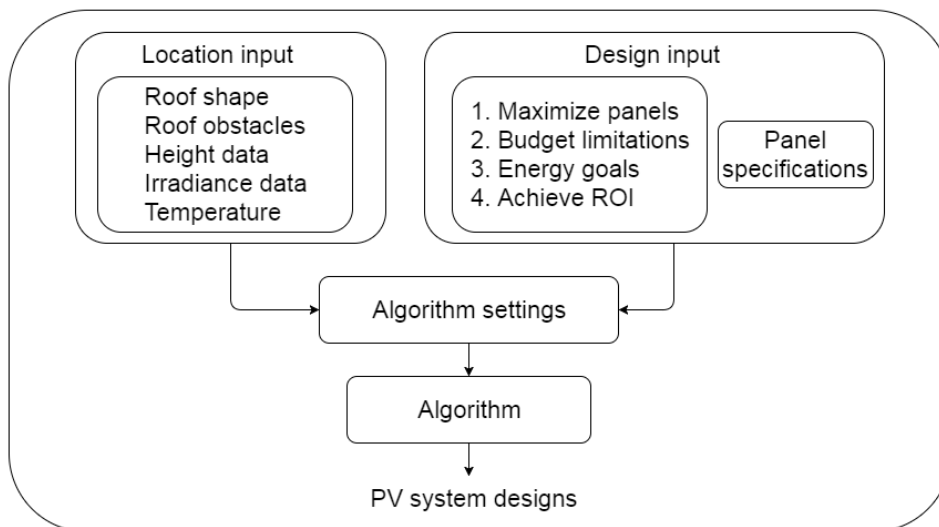


Figure 3.2: A flowchart of the inputs and the output of the algorithm that will be developed.

4

Maximum panel placement algorithms

The first algorithms that were developed attempt to fill a roof section with a maximum amount of panels. When placing a maximum amount of panels it is not relevant what the shading is on the roof. It can however be relevant if the worst panels have to be removed. For the initial panel placement the problem is just a geometrical problem. In this chapter the shading is not taken into account and the algorithm just fits as many panels on the roof as possible. Panels that are shaded a lot could be removed from the design if that is desired later.

4.1. Pitched roofs sections

For pitched roof sections, the tilt angle and the orientation of the panels is kept the same as that of the roof section. It could perhaps be possible to design a frame for the panels that would adjust the tilt of the panels to be different than the pitch angle or make the panels face a slightly different orientation. Such a construction will however be aesthetically unappealing, more expensive and will also face stronger wind loads. This is the reason that this possibility is not considered.

4.1.1. Panel placement without obstacles

The development of the panel placement algorithms was done systematically for each category of roof shape. The roof shapes that were identified are rectangle, triangle, trapezium and non standard.

Rectangle roof section

The first roof to be considered is a rectangle roof. The goal of the algorithm is to fill a rectangle with as many smaller rectangles as possible. Such a problem can be categorized as a bin packing problem. This problem has many variations and is considered often in the transport sector. It is relevant for this sector, because it is useful for determining how many items you can place on a pallet. The scenario of placing solar panels on a roof has as a restriction however, that is that the panels should either be in landscape or portrait orientation. With such a restriction the problem becomes a manufacturer's pallet loading problem [17]. In such a problem the smaller rectangles all have the same size and can rotated only by 90 degrees with the sides of the rectangle being horizontal.

Such a problem is considered a NP-hard problem [17], which is characterization in the computational complexity theory. These problems are difficult to solve, but there are algorithms that perform really well. An example of such an algorithm is the recursive partitioning algorithm. This algorithm fits the most amount of rectangles in a larger rectangle with only horizontal and vertical orientations [18]. This algorithm was tested for a simple roof example. For a roof of 820 cm by 520 cm and a panel size of 1.50 m by 0.9 m the result can be seen in figure 4.1.

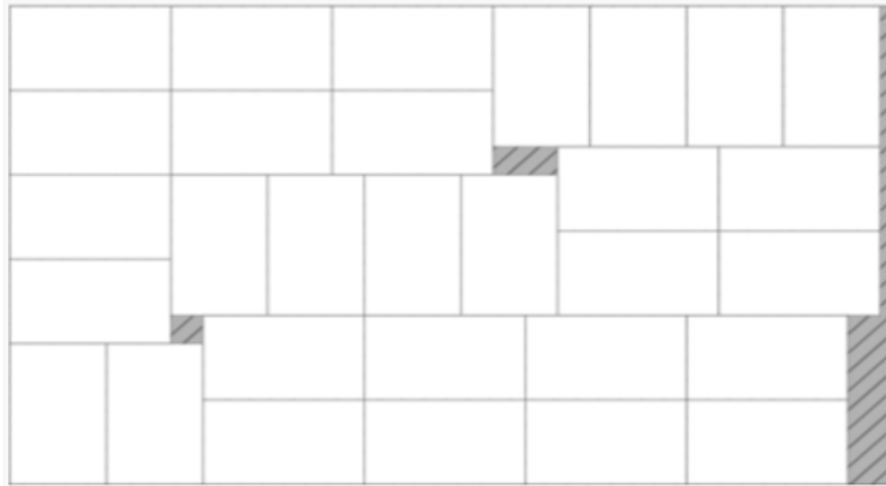


Figure 4.1: The result of the recursive partitioning algorithm placing a maximum amount of panels on a rectangle roof section. Figure obtained with a tool on a website by Birgin [19], which is based on his paper [18].

Besides it being a difficult algorithm to reproduce, there are more downsides to a solution like this. The panels don't nicely line up with each other, which doesn't look appealing. The complicated panel layout will also be troublesome for the frame that should hold the panels in place. The benefit of an extra panel on the roof due to such an exact solution is difficult to justify with these downsides. The algorithm is also not applicable to any other shape of roof.

For these reasons, it was chosen to not go for such an exact solution. Instead it was decided to develop a different algorithm. This development was done completely from scratch and the algorithm is written in python language. The results are visualized with the Python Image Library (PIL), which offers a wide variety of options to display images.

Another restriction that was added besides the horizontal and vertical orientations was the fact that the panels have to be aligned. This means that the panels have to be in rows and the panels in the same row need to have the same orientation. The simplest version of such an algorithm fits the most amount of panels on the roof with all the panels in neat rows and in the same orientation. The algorithm starts at the bottom corner of the roof and starts placing panels next to each other until no more panels fit on the row. Then it calculates how many rows would fit on top of each other and determines the total amount of panels. Afterwards it centralizes all the panels around the center of the roof surface and displays them. An added feature is that a distance to the side of the roof can be kept free of panels, which is desired when installing panels. This will effectively reduce the area of the roof that can be used. This simple algorithm was used for the same scenario as in figure 4.1, in order to compare it with the recursive partitioning algorithm. The panels as well as the roof section are treated as polygons in the algorithm. The panels that fit correctly on the roof are colored green and displayed. The results of the test can be seen in figure 4.2.

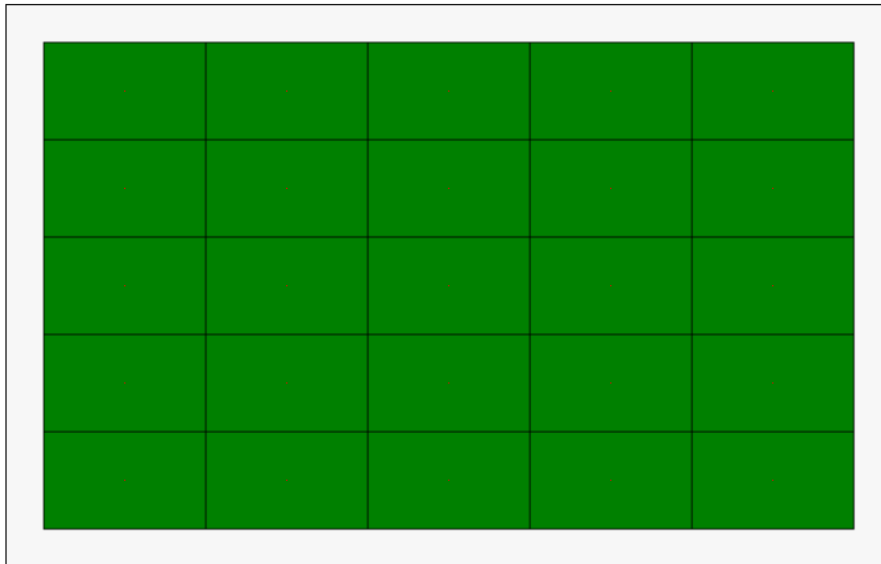


Figure 4.2: Maximum panel placement in landscape orientation on a rectangle roof section.

The total amount of panels that fit on the roof with this approach is 25, which is significantly less than the 30 that were possible with the recursive partitioning algorithm. Another attempt is made, but now with the panels in the portrait orientation. The result can be seen in figure 4.3.

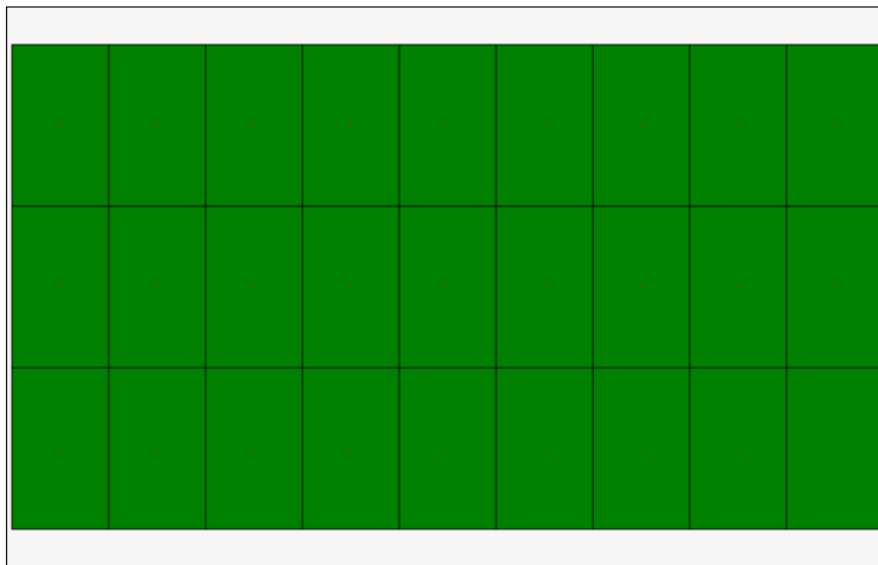


Figure 4.3: Maximum panel placement in portrait orientation on a rectangle roof section.

It matters for the total amount of panels which orientation is chosen, since now 27 panels fit on the roof. Using a single orientation is however a limitation that is not necessary. It is possible to allow for both portrait and landscape orientations on a single roof section.

A more complex algorithm has both landscape and portrait panels. It does have the restriction that either all the panels in the same vertical row have the same orientation or all the panels in the same horizontal row have the same orientation. With both orientations possible at the same time, the amount of possible solutions increases. The algorithm determines all possible linear combinations of landscape and portrait rows along the length of the roof. It then calculates how many panels can fit in each row and then sums the rows for the total amount of panels. It then does the same, but with all linear combinations along the width of the roof. For example, a certain roof can fit three landscape rows and two portrait rows along the length of the roof. Each landscape row can fit six panels and each

portrait row can fit four panels. The total amount of panels that fit on the roof is then $3 \cdot 6 + 2 \cdot 4 = 26$. This is done for each possible configuration and then the configuration with the most amount of panels is picked. The result of this more complex algorithm for once again the scenario as before can be seen in figure 4.4.

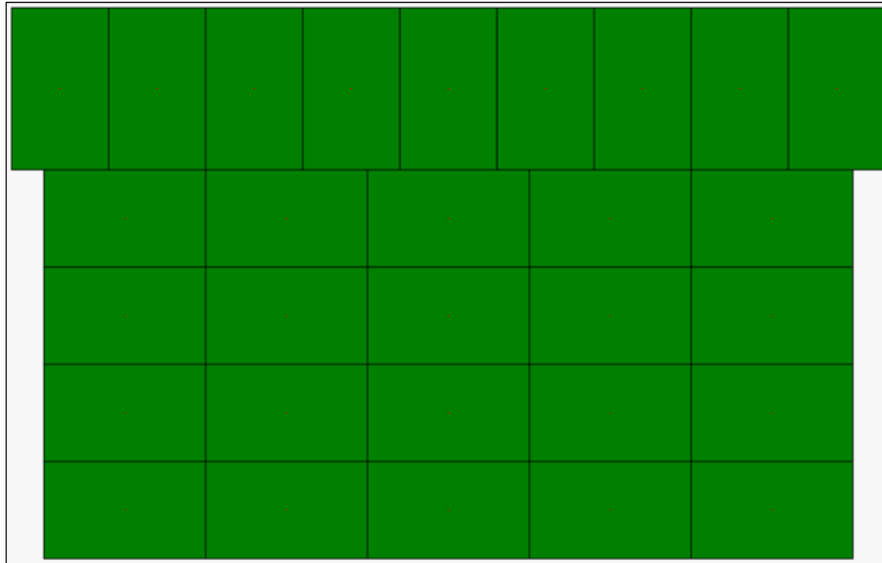


Figure 4.4: Maximum panel placement with multiple orientations on a rectangle roof section.

Allowing multiple orientations will result in the most panels in total. For this particular example the difference with the exact solution is only one panel. This self made algorithm fits 29 panels and the recursive partitioning algorithm fits 30 panels. When using this algorithm it can happen that several different solutions result in the same amount of panels. It is then be up to the installer to pick his preferred option. The panel layout looks a lot better than with the recursive partitioning algorithm, even though it fits a little bit fewer panels on the roof. The algorithm works in a fraction of a second and could give an installer a quick idea of how many panels could fit on a certain rectangle roof section without obstacles.

No existing algorithms were looked up for the next roof scenario's, because the good looking results of this rectangle algorithm encouraged the development of self made algorithms

Triangle roof section

Just like with the rectangular roof shape, the algorithm was written without any obstacles on the roof. The first algorithm had all the panel orientations either landscape or portrait. The algorithm starts filling the roof with panels at the bottom left and fits as many as can horizontally. Then it centralizes these panels and moves up one row and does the same until no more panels fit. Just as with the rectangle roof it depends on the dimensions of the roof and the panel, which orientation will yield the most panels. The algorithm determines the orientation that yields the most panels. Once again an edge that has to be kept free can be entered. This edge is represented by a red line. This edge could be different for each side, or can be the same for all sides. A visualization of a triangle filled with this algorithm can be seen in figure 4.5.

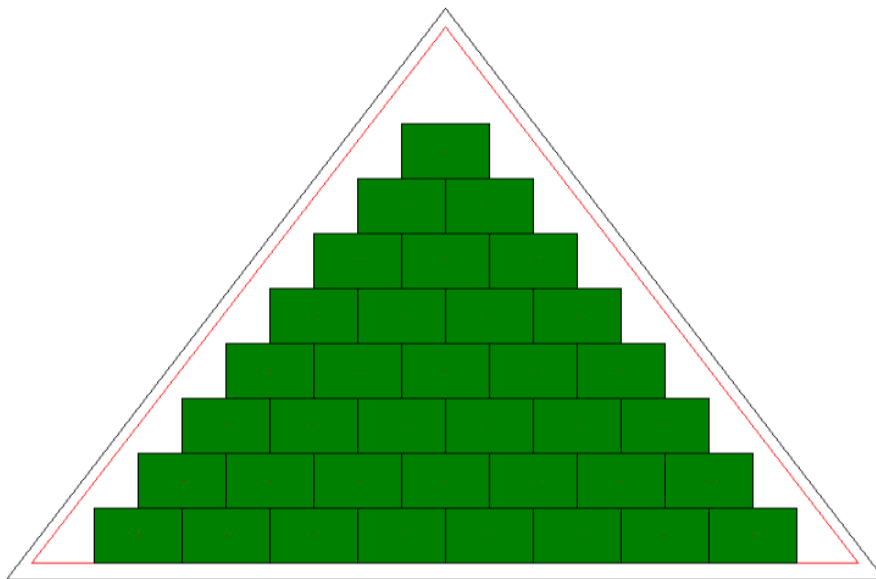


Figure 4.5: Maximum panel placement with a single orientation on a triangle roof section.

Just as with the rectangle roof surface, using only a single orientation of panels might look a bit better, but will not always result in the most amount of panels. The algorithm for a triangle roof section is expanded by also including the option of having multiple orientations. Due to the fact that the roof is not the same width at all places, this expansion is more complex than with a rectangle roof section. When placing the landscape and portrait rows on top of each other, it didn't matter what the order was with the rectangle roof section. The order is however of great importance with a triangle roof section. The first step is now to find all a linear combinations of landscape and portrait rows along the height of the triangle.

possible scenario could be that four landscape row and two portrait rows fit along the height of the triangle. It matters in which order these rows are placed, because the width of the roof surface is not constant. This results in a lot of permutations. All permutations are tried and the algorithm counts how many panels will fit in each row and then sums all the rows to get the total amount of panels. The order of panels which results in the most amount of panels is then determined and chosen as the final configuration. The result of this algorithm for the same situation as figure 4.5 is shown in figure 4.6.

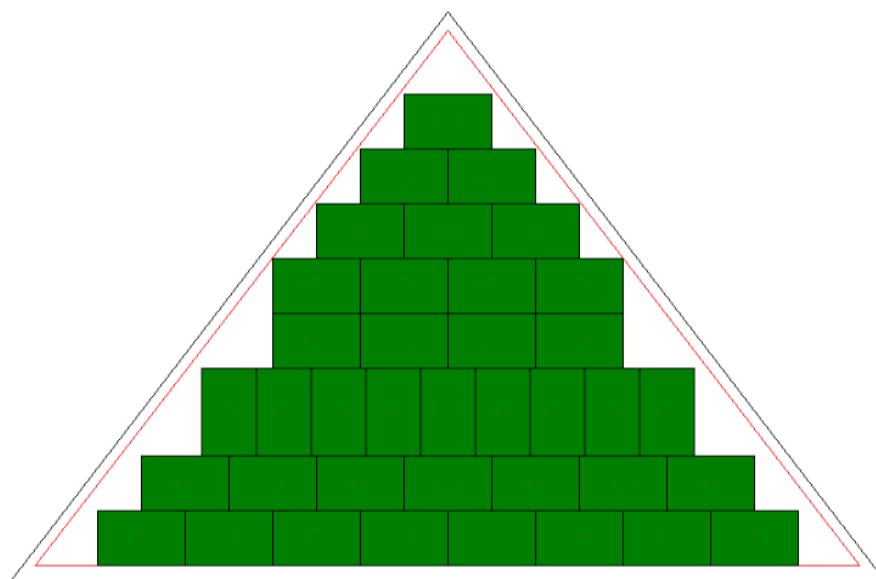


Figure 4.6: Maximum panel placement with multiple orientations on a triangle roof section.

When using multiple orientations the total amount of panels is more than with just a single orientation. The amount of panels in figure 4.5 is 36 and in figure 4.6 is 38. The algorithm can be used to know within a second how many panels would fit on a triangle roof section without obstacles.

Trapezium roof section

A trapezium roof section is very similar to triangle roof section. The algorithms that were developed work the same as for a triangle. The difference is only that at the top the roof section doesn't run into a sharp peak, but that doesn't affect the workings of the algorithm. The result of the algorithms with a single orientation and multiple orientations can be seen in figure 4.7, with the red line once again indicating an arbitrary edge that has to be kept free.

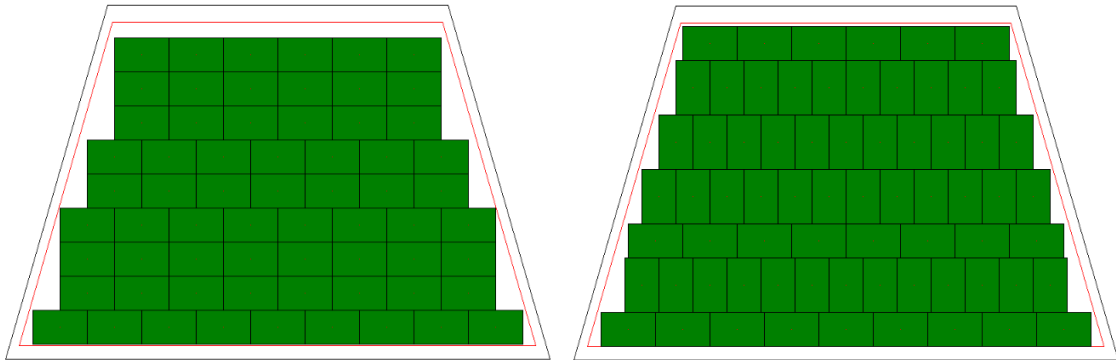


Figure 4.7: Maximum panel placement with single or multiple orientations on a trapezium roof section.

Multiple orientations can result in more panels on the roof. The solution on the left has 65 panels and the solution on the right has 69 panels. This algorithm can be used to determine the amount of panels that fit on a certain trapezium shaped section within a second.

All of these previous roof sections are in a perfectly defined shape. The algorithms created for those situations do work well in placing the most amount of panels. In reality the roof sections that will be used will not always have such a perfect shape. This means that an algorithm should be able to work for any shape. Such an undefined roof section is called a non standard roof shape and those are explored next.

Non standard roof section

None of the previously made algorithms can be used for a non standard roof section. An entirely different method was developed instead. The method that was developed starts by placing a grid of polygons with the size of the panels on top of the roof surface. Each polygon represent a panel. A grid for landscape and portrait orientation can be made and examples of those can be seen in figure 4.8.

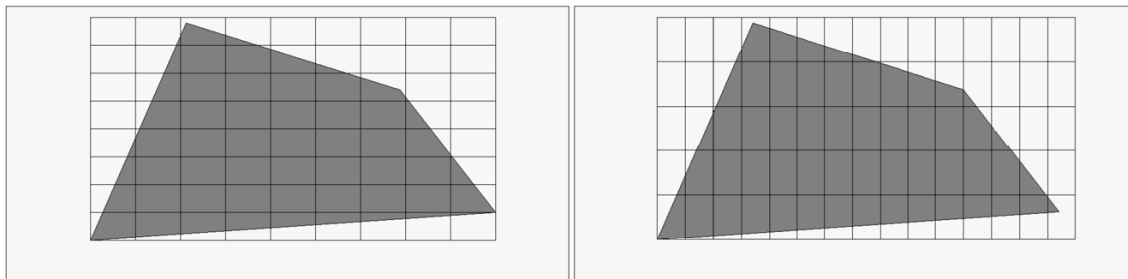


Figure 4.8: Placement of a panel grid on a non standard roof surface. Landscape orientation on the left and portrait orientation on the right.

The size of the panel grid is determined by the bounding box of the polygon of the roof section. The bounding box is the smallest rectangle that fits around the entire polygon. The bottom left of the

bounding box is used as a starting point and then the length and width of the entire grid is determined with the distance to the top right of the bounding box. The smallest grid that entirely covers the roof section is then created. Once the grid is formed, the next step is to determine which panels fit on the roof. This is done with the Shapely library in python, this is a library that enables the manipulation and calculations with polygons. It has a function that checks whether a certain polygon is inside another polygon. This is used to check for all the panels whether they lie within the roof polygon. If a panel fits on the roof it is displayed in green, which is illustrated in figure 4.9.

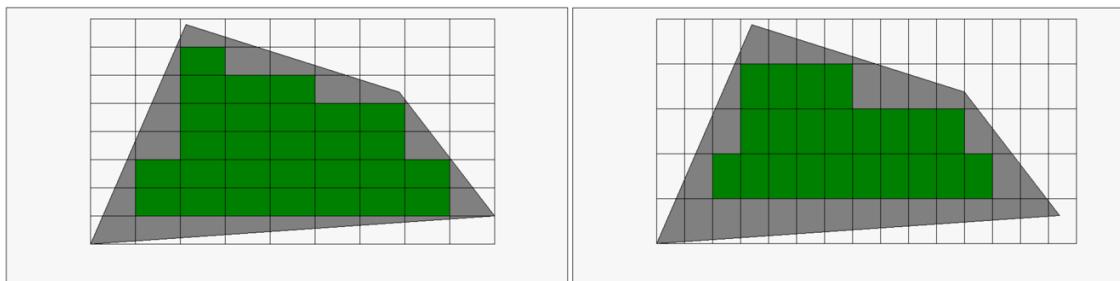


Figure 4.9: Panels that fit within the roof polygon are colored displayed in green.

The final step is to remove all the panels that don't fit on the roof. This is done by saving only the panels that fit within the roof polygon into a new list. The final result for both landscape and portrait orientation is shown in figure 4.10.

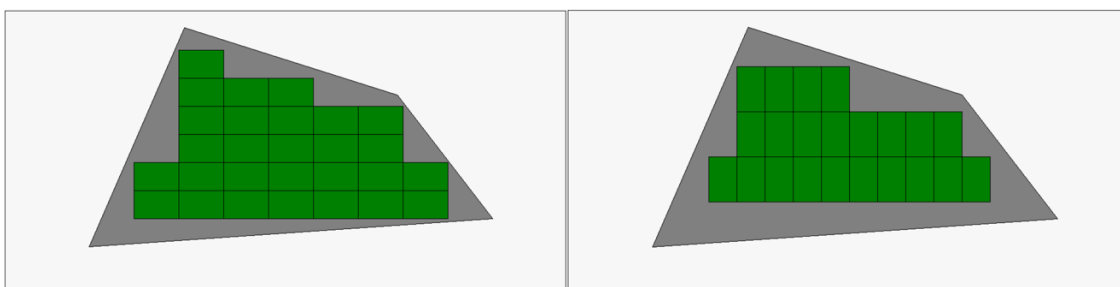


Figure 4.10: Panels that don't fit within the roof polygon are removed.

This is a very fast way of determining how many panels fit on a non standard roof section. When filling the roof like this, it matters at which point the grid of panels starts. To maximize the amount of panels on the roof the grid is shifted across the roof and after each shift the amount of panels is calculated. The grid shifted vertically in a certain amount of steps until it has shifted one time the height of a panel. It then starts at the bottom again with one step shifted horizontally. This continues until a total horizontal shift has take place that equals one width of a panel. This ensures that all ways of placing the grid are checked. The step size of this shifting process will determine the speed and accuracy of the algorithm. This algorithm can be used to determine within seconds how many panels of a certain orientation will fit on a roof section of any shape without obstacles.

4.1.2. Panel placement including obstacles.

The previous scenario's all had no obstacles on the roof. When obstacles on the roof are taken into account, such as windows and chimneys, the problem becomes more complex. The previously developed algorithm don't work in this scenario.

Rectangle roof section

A first idea was to divide the roof section into pieces without obstacles, which would enable an algorithm without obstacles for that particular section. An example of dividing a roof section into smaller roof

sections can be seen in figure 4.11.

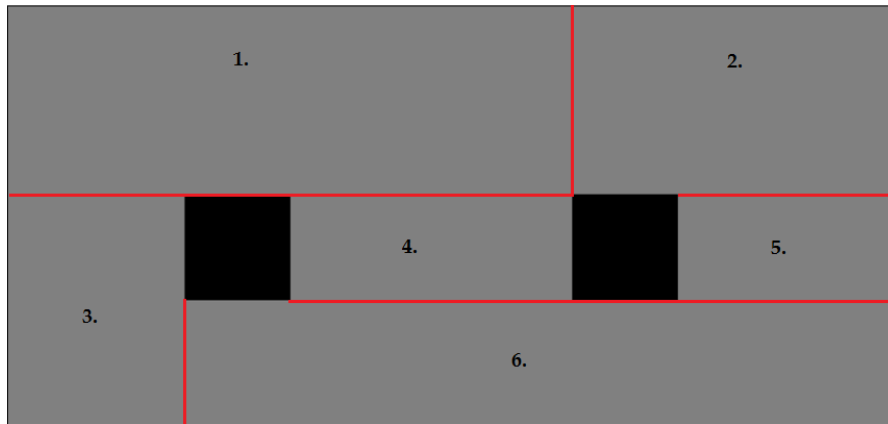


Figure 4.11: A rectangle roof with obstacles divided into sections without obstacles.

In this figure the black rectangles represent obstacles. The roof section is divided into six rectangular sections that do not include any obstacles. There were however immediate challenges with using an approach like this. Dividing the roof into these rectangular sections can be done in many different ways, it just depends on how the lines are drawn. Big questions are how to draw these lines and how much it matters how these lines are drawn. There is also a second problem with this method. This problem is that filling in the obtained rectangle sections independently from each other will lead to a misalignment of panels at the boundaries of the sections. This makes mounting the panels more complicated and it also is aesthetically unappealing. In order to prevent this, the filling of the sections should not be independent. They would have to be coordinated and that would become a complex algorithm. These issues prompted the search for an alternative approach.

This second approach does not divide the roof into sections, but instead ignores the obstacles in first instance and treats the roof as if no obstacles are there. After the algorithm is done, the panels that overlap with the obstacles are removed. An example of a result from this method can be seen in figure 4.12.

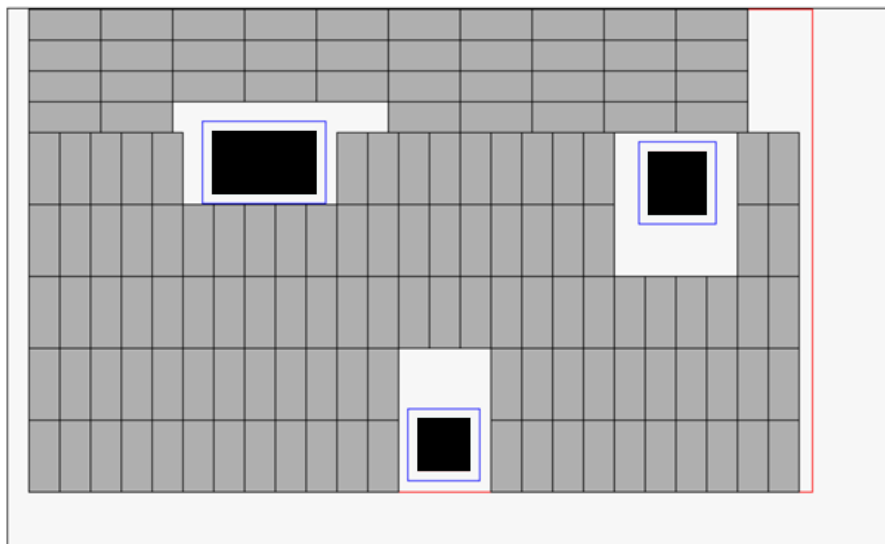


Figure 4.12: Maximum panel placement for a rectangle roof with obstacles.

The red line indicates the edge to the roof that has to be kept free. The blue lines around the obstacles indicate the distance towards the obstacle that has to be kept free. Both these distances were arbitrarily chosen in this case. A big benefit of this method is that there is no misalignment of panels.

Even though most of the available area is covered, some extra panels could however still be squeezed in. Sometimes a row of panels can be shifted in order to fit in an extra panel. In the example in figure 4.12 this is also the case. The algorithm detects which rows could be shifted in order to make room for extra panels. The top four rows were shifted to the right in order to create space for an extra panel. This is visualized in figure 4.13.

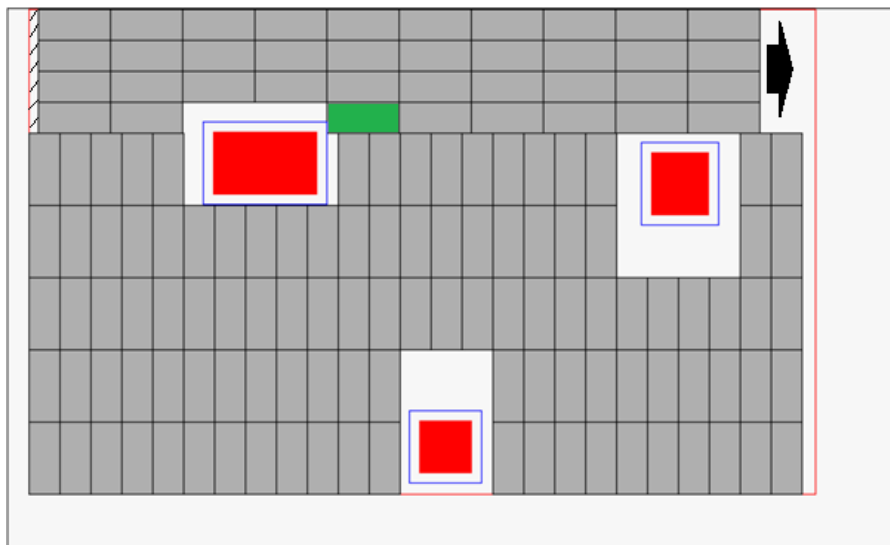


Figure 4.13: Maximum panel placement for a rectangle roof with obstacles with extra shifts to place more panels

In this case only one row had to be shifted in order to create the necessary space, but all four rows were shifted to keep the symmetry. When this approach is used it can happen that several rows can be shifted to add panels. These shifts can be of different lengths, which means that the symmetry of the layout will be lost. It is up to the installer to decide whether adding extra panels in that fashion is desired. Another method to include more panels is to fill some of the remaining area near to the obstacles. The panels that could fit in this space are in a different orientation than the surrounding panels as can be seen in figure 4.14.

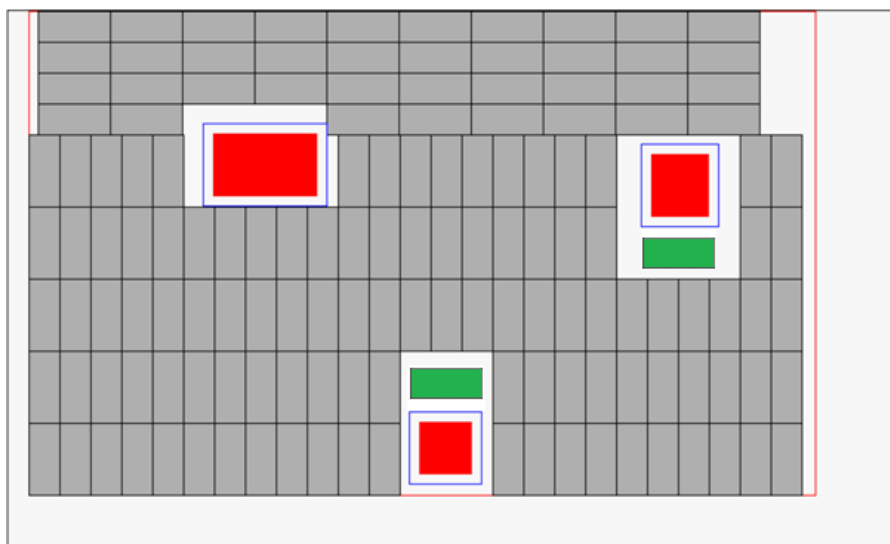


Figure 4.14: Placement of extra panels that have different orientation than nearby panels.

Placing panels in this manner will increase the total amount of panels, but it is not as aesthetically appealing and it also creates some additional effort when mounting the panels on the roof. This panel

placement algorithm can also be applied to the triangle and trapezium roof with some adjustments. The rows can then only be shifted horizontally to create room for extra panels.

Non standard roof section

The algorithm that was used for maximal panel placement for non standard roofs without obstacles only changes slightly. Instead of only checking whether the panel polygons lie within the roof polygon, now it is also checked that they do not intersect an obstacle polygon. This is once again easily done with the Shapely library. A result of this can be seen in figure 4.15.

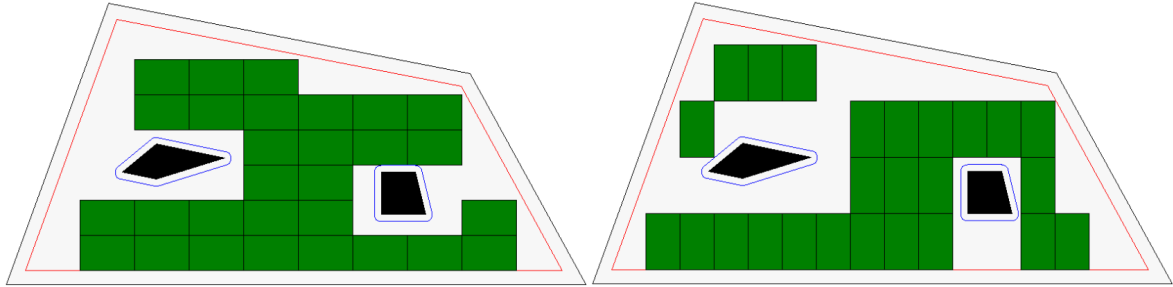


Figure 4.15: Maximum panel placement for a non standard roof including obstacles.

This is the algorithm that can be applied to all roof sections and it will place the most panels in a single orientation. It is sometimes possible to place a few more panels, but this is not done for aesthetic and panel mounting reasons.

4.2. Flat roofs

Placing the panels on a flat roof is different than for a pitched roof. These differences arise due the fact that the tilt angle and the orientation of the panels is not fixed by the roof for a flat roof. Since the panels are not just placed directly on the roof there is also the aspect of self-shading. These aspects and how it is incorporated in the algorithm are discussed in this next section.

4.2.1. Optimal tilt angle

The tilt angle has an impact on the performance of the panel. The optimal tilt angle depends on the latitude on earth where the panels are positioned. In some instances it could be beneficial to adjust the tilt angle of the panels for different seasons. This option was however not considered in this thesis. A lower tilt angle than would be optimal is usually used, because this will reduce the shadows created by the panels and enables more panels to be placed efficiently on the roof. Some of the performance of each panel is given up in order to place more panels, which results in a higher total yield.

4.2.2. Self-shading and row distance

When placing panels on a flat roof the panels have to be mounted on a rack that fixes the angle. The panels will create a shadow behind it. If the rows of panels are placed too close to each other this shadow can hit the other panels. This effect is called self-shading and is visualized in figure 4.16.

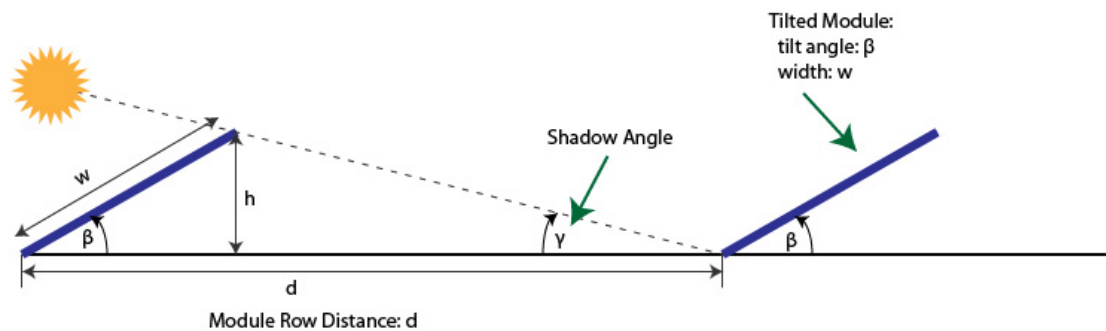


Figure 4.16: The shadow caused by a panel on flat roof. Figure taken from [20]

The tilt angle is of importance for the algorithm for two reasons: it determines the row distance and it determines the ground area covered by the panel. The row distance is also needed to access the panels. With the pitched roofs the actual panel size could be used as the area covered by the panel, but that is not the case for tilted panels on flat roofs. The area that is actually covered by the panel is smaller than the width of the panel. The distance covered by the panel is called L and is given by:

$$h = w \cdot \cos(\beta). \quad (4.1)$$

The size of the rectangles that represent the panels use L as one of the edges. The exact relationship between the tilt angle and the row distance is not derived, as it is not important for this stage of the development. The values of the tilt and the row distance are adjustable in the algorithm. A derivation of the tilt angle and the row distance can be added later and then the algorithm will work with the derived values.

4.2.3. Orientation of the panels

While a pitched roof offers no room to vary the orientation of the panels, a flat roof has offers all possibilities. In this project three different orientations are explored: the optimal orientation, east-west orientations and an orientation in line with the orientation of the roof surface.

Optimal orientation

For the northern hemisphere, and thus the Netherlands, a south orientation results in the most total energy produced. There is however a bit more to think about than just the total amount of energy. Placing a solar panel in a different orientation will not only result in a lower total energy yield, but the bulk of energy will also be produced at a different point during the day. This could be interesting when combined with the times of the peak load. Not letting all the panels face south could also be good for power grid considerations. This paper is only focused on single roofs and these aspects are beyond the scope of this research.

The LiDAR data can give the extra benefit of more optimally determining the best orientation of the panels. This is due the fact that the surrounding obstacles can be taken into account. A south orientation might no longer result in the maximum energy yield, if there is a huge obstacle south of the roof. If there is a huge obstacle in the south direction, a change of panel orientation could be a good option. Such a scenario can be seen in the sketch in figure 4.17.

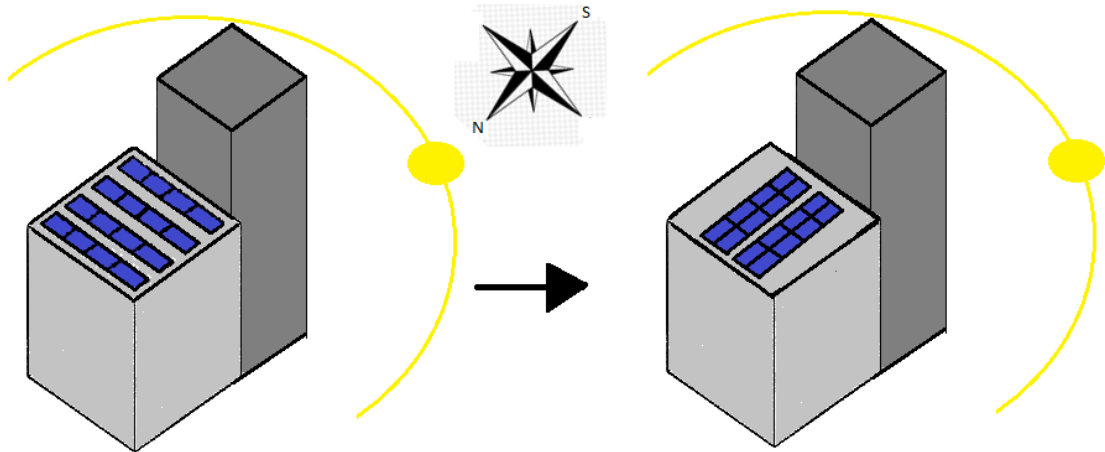


Figure 4.17: A south orientation is changed to an east-west orientation, because of large obstacles in the south.

Obstacles such as in this scenario can be good reason to choose for an east-west orientation, even if that wasn't considered at first. It is also possible that a south orientation is slightly adjusted due to an obstacle. Such a situation is depicted in the sketch in figure 4.18.

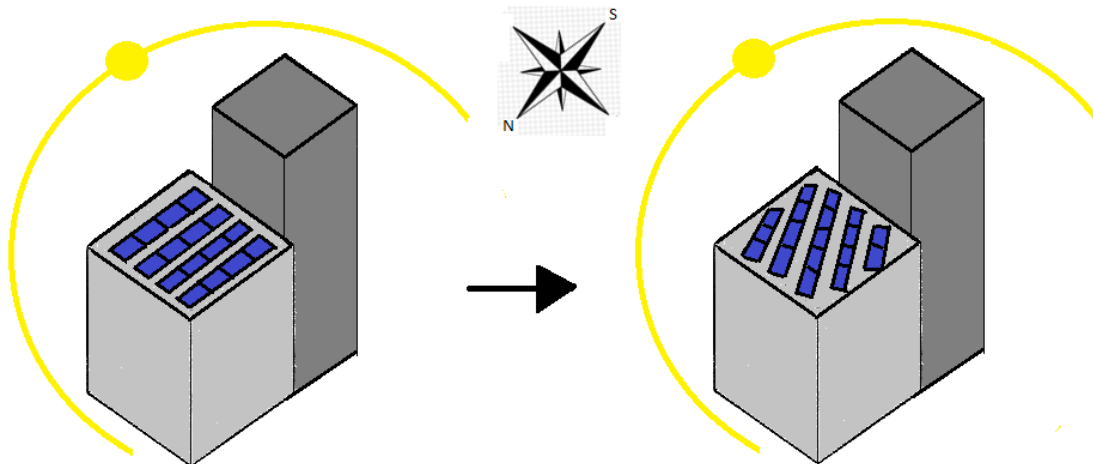


Figure 4.18: A south orientation is adjusted to a south-south-east orientation due to large obstacles in the west.

This obstacle in figure 4.18 will block a lot of the sunlight coming from the west, so it might be better to rotate a bit to the east in order to receive more sunlight from the east. This concept could result in a higher total yield. It will especially be the case in cities with large buildings that are very near each other. This isn't much this case in the Netherlands, but the concept is still interesting.

A quick test was done with the Solar Monkey software in which the energy yield was calculated for different orientations with a large obstacle near the roof. The location tested was a roof near the Electrical Engineering, Mathematics and Computer Science (EEMCS) building in Delft. A view of the obstacles created with the Solar Monkey software can be seen in figure 4.19.

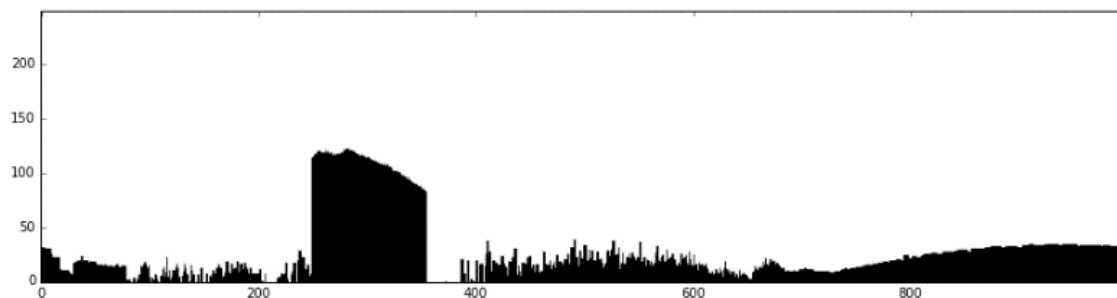


Figure 4.19: The obstacle view for a roof near the EEMCS building.

The obstacle view is constructed with the LiDAR height data and shows the surrounding objects around a point up to certain distance. The obstacle view is explained in more detail in chapter 7. The EEMCS building can clearly be seen in the south-east direction. The performance of a single panel was calculated for different orientations. One time it was done including the obstacles and one time without the obstacles. Two graphs are created that show the performance at each orientation. The graphs are normalized and can be seen in figure 4.20.

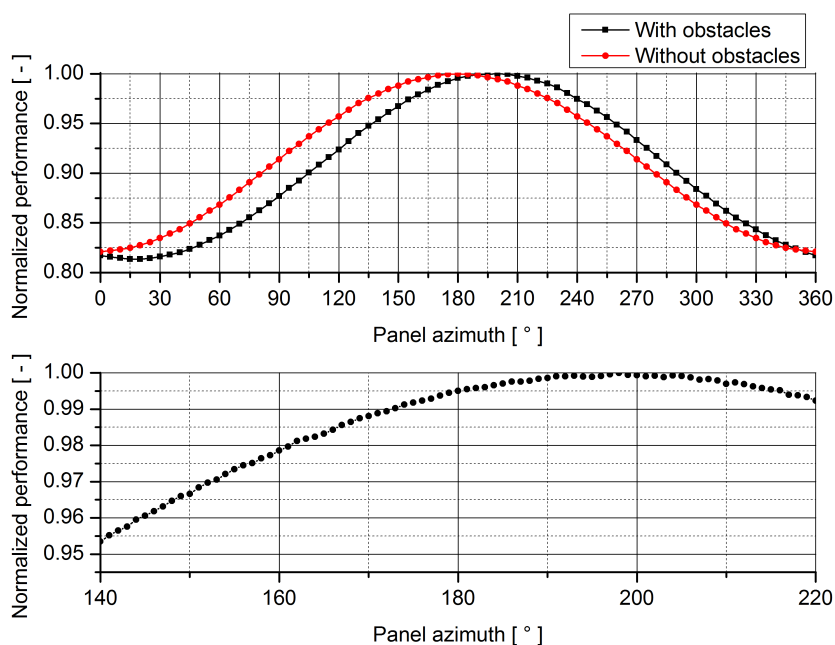


Figure 4.20: The normalized performances of a panel with a tilt angle of 12 degrees and various azimuths, with and without the surrounding obstacles included in the algorithm.

The performance without obstacles is the highest when the panel is facing south, which is to be expected. The optimal orientation however shifts when the obstacles are taken into account. This shift is towards the south-west, which can be explained by the large building in the south-east. This graphs shows that the inclusion of surrounding obstacles in the determination of the optimal azimuth can make a difference. It is however only a difference of about 0.5%. That could however still be significant for large systems in a big city. This test was done to test the theory that surrounding obstacles influence the optimal orientation, it is not analyzed any further in this thesis.

East-west orientation

A different set-up than facing the panels towards the optimal orientation is an east-west orientation. A big benefit is that there is no self-shading, because the panels are placed back to back. This means

that more panels will fit on the roof. Another benefit can be that the output of the system is more distributed throughout the day, instead of having a peak output at noon.

Panel orientation in line with roof orientation

Another option that is frequently seen is to place panels in the same orientation as the roof. This means that for a flat rectangle roof that is rotated with the long side facing west, that the panels will also face west. This will be made more clear in the upcoming section. This option will often result in the most panels on the roof. It usually results in a lower performance per panel than choosing a south orientation, but the extra panels that fit can make it still the preferred option. All of the different panel orientations were considered in the development of the algorithms.

4.2.4. Maximum panel placement including obstacles

For the development for this algorithm a lot of inspiration was taken from the non standard pitched roof maximal panel placement algorithm. The same algorithm is applied with a few changes. These changes are a correction for the effective panel area, a row distance that has to be maintained and the orientation of the panels can change. This first change is done by simply changing the size of the polygons used as panels and the other change is done by changing the way that the panel grid is created. The panel polygons are no longer all created next to each other, but the next row is formed a row distance apart from the previous row. The row distance is different depending on whether the panels are placed in landscape or portrait orientation. The orientation can be chosen and the panel grid then rotates to face the right direction. An example of the algorithm applied to a flat roof with some obstacles can be seen in figure 4.21.

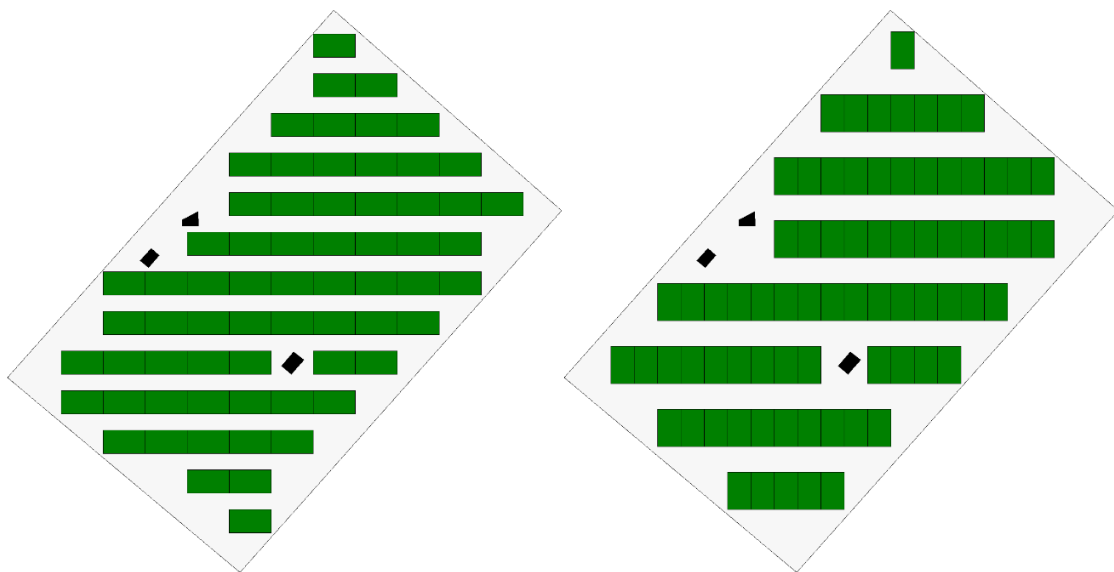


Figure 4.21: Maximum panel placement for a flat roof with obstacles for both landscape and portrait orientation with the panels facing south.

Placing the panels in line with the roof can be done in two ways (four actually, but a 180 degrees difference doesn't matter geometrically). The first step to implement this is to determine the orientation of the roof. With manual designing this is done by rotating the panels until they align with the roof. The algorithm should however automatically determine the orientation.

A method is developed that is based on the angle of the longest side of the roof polygon with the north-south axis. This is done by determining all the lengths of the sides of the roof polygon, picking the longest side and calculating its angle with the north-south axis. An illustration of this can be seen in figure 4.22.

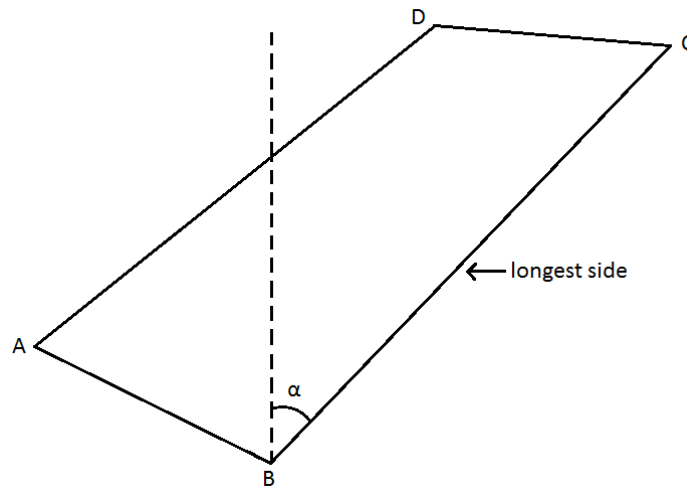


Figure 4.22: The angle α that defines the roof orientation.

The angle α is then given by

$$\alpha = \arctan \frac{x_2 - x_1}{y_2 - y_1}. \quad (4.2)$$

This method for determining the orientation of the roof is also evaluated and the results can be found in chapter 9. The pseudo code for determining the orientation is given by algorithm 1 in Appendix B.

There two ways to place the panels aligned with the roof and the version is chosen that has the panels closer to facing south, which will in general yield a higher performance per panel. Both versions of maximum panel placement with the orientation in line with the roof is anyway shown in figure 4.23.

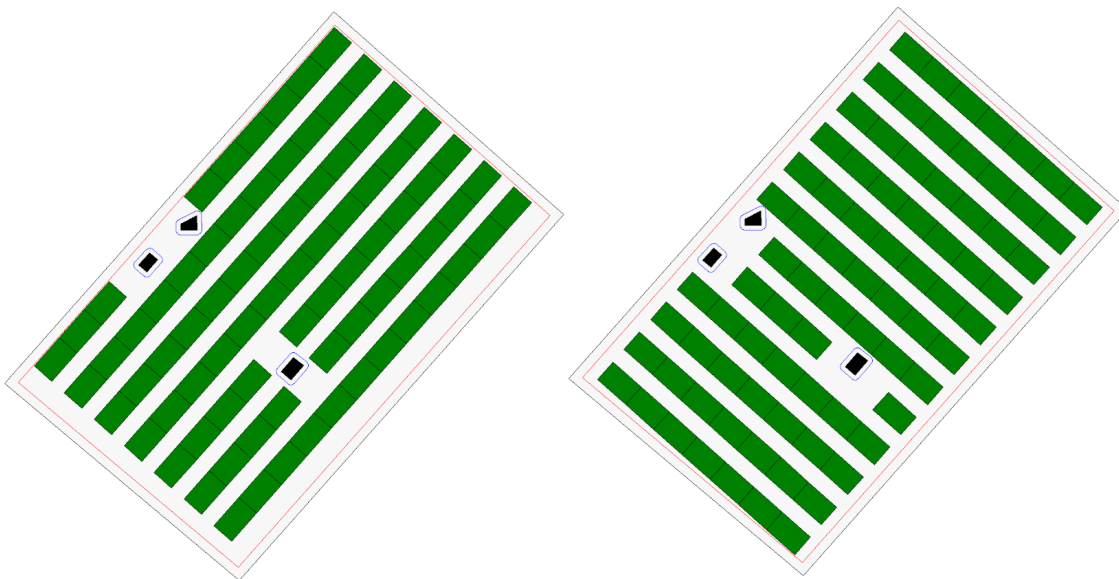


Figure 4.23: Maximum panel placement for a flat roof with obstacles with panels orientated in line with the roof in two different ways.

The amount of panels that fit on the roof will depend on which version of orientation is chosen. The orientation will also have an effect on the performance. To get the best total performance both scenario's should be calculated.

An east-west orientation has two rows of panels placed back to back. This requires another adjustment in how the panel grid is created. The panels can also be placed back to back, but then orientated

in line with the roof, which can once again be done in two ways. All back to back versions are shown in figure 4.24.

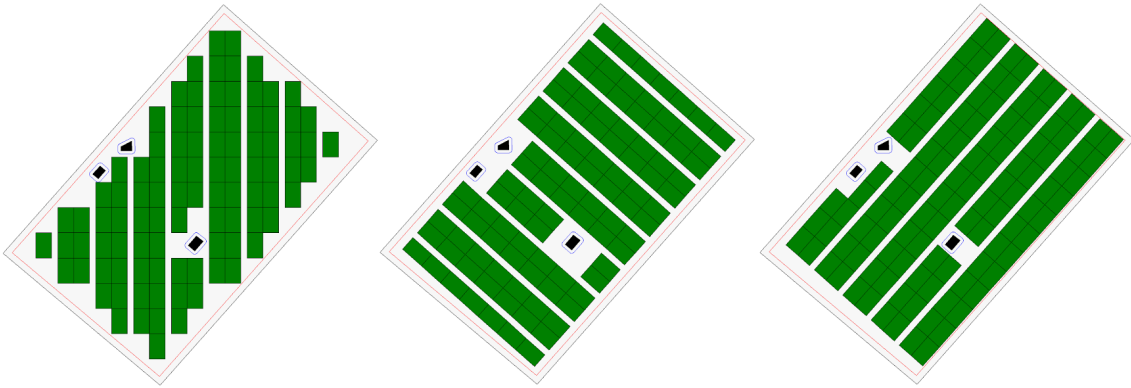


Figure 4.24: Maximum panel placement for a flat roof with obstacles with panels back to back in three different versions.

Each back to back version will result in a different amount of panels. This will once again lead to a decision that has to be made based on the average performance per panel and the total amount of panels that can be placed.

All of these maximum panel placement algorithms work within seconds and are useful to determine how many panels fit on the roof. The algorithm is evaluated in chapter 9. The algorithm for filling a roof with a maximum amount of panels is given in pseudo code by algorithm 2 in Appendix B.

5

Finite panel placement

5.1. Introducing shadows

Filling the roof with a maximum amount of panels is not always desired. The amount of panels can be based on a certain budget or a certain energy requirement. This means that only up to certain amount of panels are placed and this might cover only a portion of the available roof area. In such cases it becomes relevant what kind of shading occurs on the roof surface. Some areas on the roof will be better than others, due to the amount of shading it receives. The first step in developing an algorithm that takes the shading into account, is to determine how this shading on the roof is defined. This is explained by starting out with a very simple sketch of a rectangle roof section that contains two obstacles and three shaded areas. This sketch can be seen in figure 5.1.

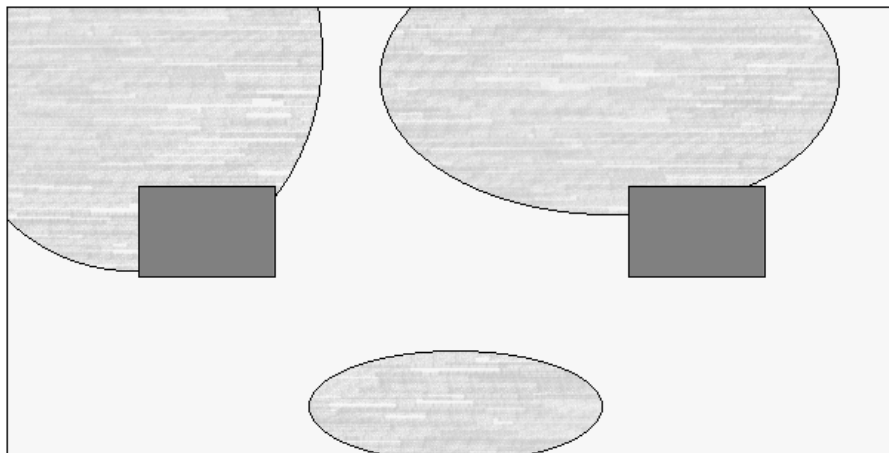


Figure 5.1: An artificial rectangle roof surface with two obstacles and three shaded areas.

The three shaded areas could have been caused by obstacles next to the roof, such as a tree. The next step is to determine how these shadows are represented. If all areas on the roof have a certain score that represents how good that position is, then these shaded areas should get a low score. This grading of the areas is introduced in the next section.

5.1.1. The PV performance grid

In this research a shadow analysis is performed with LiDAR 3D data and an algorithm developed by Solar Monkey. A grid with squares of 0.5 by 0.5 meters is created and in the middle of each grid point the performance of a single panel is calculated. This performance value is then assigned to that grid square. This will result in a grid of performance values with a resolution of 0.5 by 0.5 meter. This

grid resolution is chosen, because it is the same resolution as the LiDAR height data on which the performance calculation is based.

During the development of the algorithm this was however not yet done and artificial values were used instead. These values range from zero to one, in which one represents an area without shading and a zero an area that is shaded all the time. An example of such a grid, based on the roof of figure 5.1 can be seen in figure 5.2.

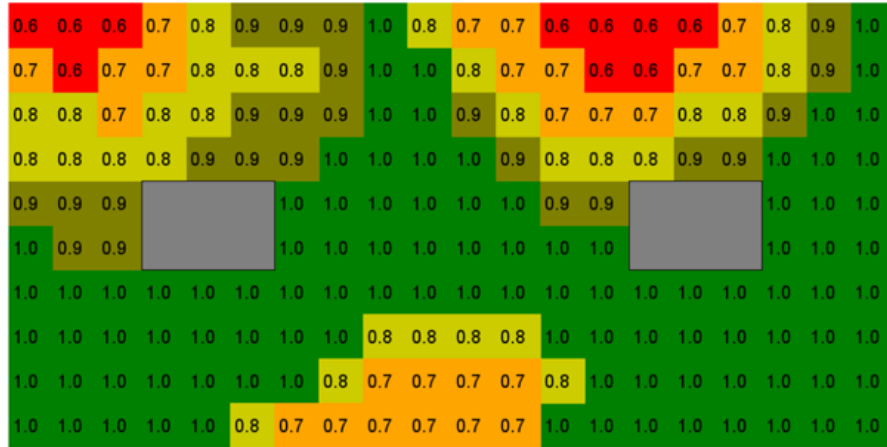


Figure 5.2: An artificial performance grid for the roof of figure 5.1.

This grid consists of square polygons with a dimension of 50 cm by 50 cm. Each of those polygons has a corresponding value that represents the amount of shading it receives throughout the year. In later testing on actual roof scenario's these shadow values are replaced by calculated performance values. In order to use this grid for optimal panel placement, the performance of a panel placed on the grid has to be determined. The performance of a panel is discussed in the next section.

5.1.2. The performance of a PV modules

The PV modules are the core of a PV system. These are the devices that will generate the electricity. To understand the importance of shading on the panels, the basic workings of a PV module is discussed. A PV modules has the most output if they operate at a certain voltage, this point of operation is called the Maximum Power Point (MPP) and the corresponding voltage is called the Maximum Power Point Voltage (V_{MPP}). The power output of the PV module is determined by multiplying the voltage and the current. The IV curve shows the voltage and current for a single PV module, an example is shown in figure 5.3.

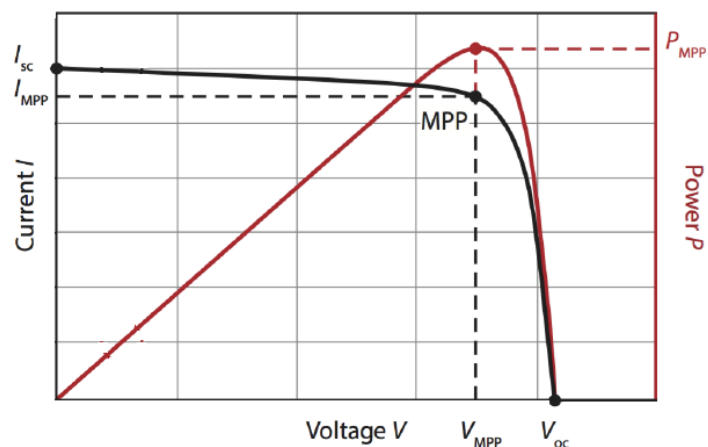


Figure 5.3: The IV and PV curve of a PV module. Figure taken from [8].

The current of the PV module is determined by the incoming solar irradiance. The voltage and the current follow a specific curve. The operation power of the PV module can be controlled by changing the voltage, which can be done by changing the load. The load on the panel can be controlled in such a way that the panel operates at its maximum power point, this is done by a maximum power point tracker (MPPT).

The performance of a PV module is listed by the manufacturer and the PV modules are tested under standard testing conditions (STC). The STC have an incident irradiance of 1000 W/m^2 with an air mass (AM1.5) 1.5 spectrum at a cell temperature of 25 degrees Celsius [8]. All of these aspects have an effect on the performance of the panel. The AM1.5 spectrum defines how the solar spectrum that hits the panel looks like, the irradiance determines how many photons are hitting the panel and the ambient temperature has an effect on the temperature of the panel. A higher temperature of the panel has a slight positive effect on the current, but a large negative on the voltage. The effects of an increased irradiance or an increased temperature can be seen in figure 5.4.

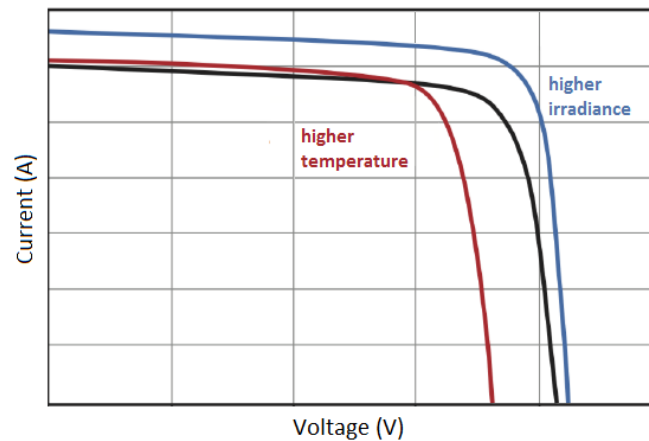


Figure 5.4: The effect of irradiance and temperature on the IV curve of a PV module. Figure adapted from [8].

Due to changing ambient conditions the IV curve will change continuously. This means that the MPP changes a lot and the MPPT will continuously adjust the operating point. The manufacturer will give certain percentage efficiency loss per temperature increase for the panel on the specification sheet. The curves in figure 5.4 show that having a higher irradiance is good for the performance of the panel. Avoiding the shadows on the roof as much as possible will result in a higher performance.

The first step to develop the algorithm is to determine how to assign a certain performance to a panel that is placed on the performance grid. In order to determine the performance of a panel it is necessary to dive a bit deeper into what the solar panel consists of. A typical crystalline silicon panel consists of multiple solar cells that are connected in series. A very typical example of this can be seen in figure 5.5.

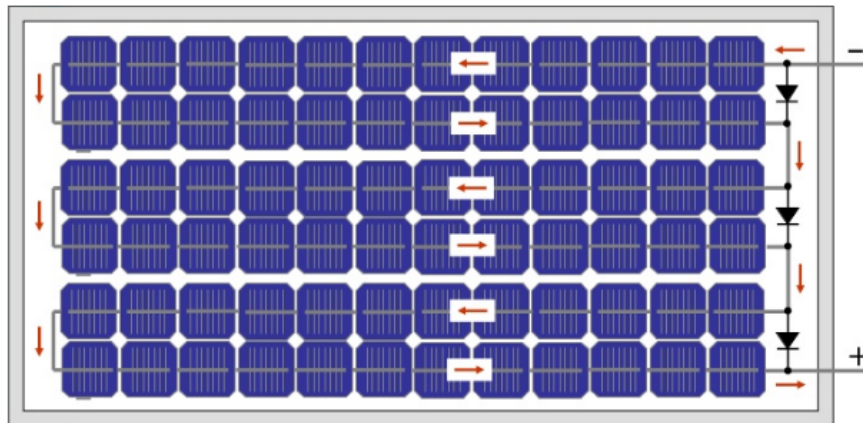


Figure 5.5: An example of a solar panel with the cells visualized. The diodes on the right side are called bypass diodes. Image is from [21]

Under normal operation the current flows through all the cells in the direction of the arrows, without going through the bypass diodes. This panel consist of 72 solar cells connected in series. Each solar cell will generate a current depending on the incoming irradiance. The current production of a solar cell will decline significantly if it is shaded, due to reduction of incoming solar irradiance. The current of a string of solar cells connected in series is limited by the solar cell that produces the lowest current [8]. Since all solar cells are connected in series, it means that if a single cell is shaded, it will have a detrimental effect on the performance of the entire panel. The bypass diodes are installed to reduce this effect. If one of the cells is performing bad, the current flows through the bypass diode instead of through the string with the bad cell. This basically turns of that string. The result is that the current in the panel is no longer limited by this bad cell. The result is that the overall performance of the panel will be better, because the cells in the other stings can still contribute to the power output.

The size of a typical solar panel is about 1.6 m by 1.0 m, which is larger than the resolution of the performance grid. This means that each panel will overlap with multiple squares of the performance grid. This results in multiple possibilities of assigning a performance value to a panel. Three methods are briefly discussed here.

In the most simple case a single value is assigned to the entire panel. In this method the center point of the panel determines the value for the entire panel. An example of how this works is given in figure 5.6, followed by an explanation.

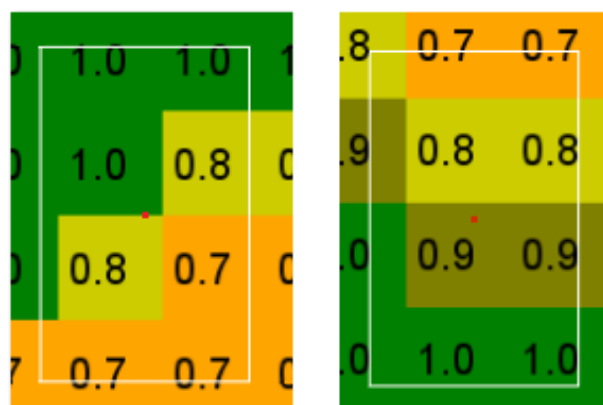


Figure 5.6: Two panels on top of the performance grid, with the red dot as the center point of the panel.

If the center point of the panel is on top of a boundary of two or more performance grid squares, the average of those values is taken. That means that the left panel in figure 5.6 will get the performance

value of 0.9.

A slightly more precise method is to use three values. Usually a panel consists of a long string of solar cells with three bypass diodes that are placed at one third of the panel, as is the case in figure 5.5. Dividing the panel into three horizontal sections can result in a more precise prediction of the yield of that panel. In partially shaded conditions it is more precise to have a performance value for each of the three sub strings in the panel. The panel will then be divided into three sections along the length of the panel and a center point of each section will be determined. An example of how these panels would be divided can be seen in figure 5.7.

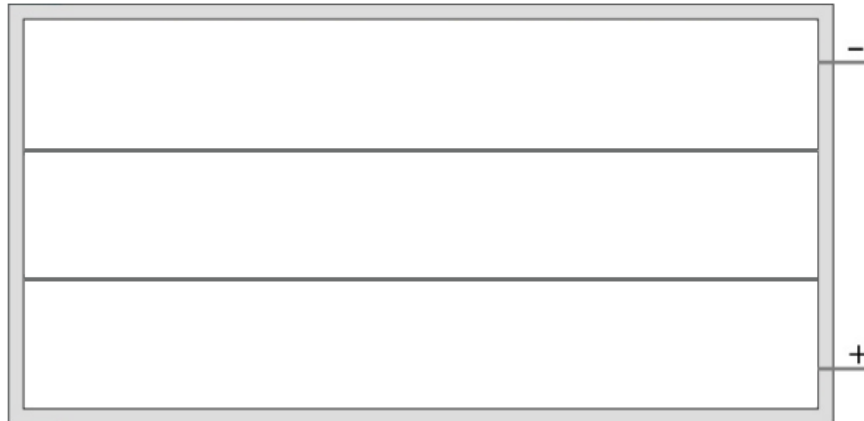


Figure 5.7: Dividing the panels into three horizontal sections.

The values of each of the three sections are determined in the same way as is done with the method for a single value per panel.

The third method is to use interpolation to obtain a value for each cell in the solar panel. With knowledge of how the cells are positioned in the panel an interpolation can be done to gain a value for each solar cell.

The resolution of the LiDAR data however determines how accurate these methods are. The resolution is 0.5 by 0.5 meters. The last two options use a resolution that is higher than this resolution and are therefore deemed unfit. Using those methods would claim more precision than is actually present. It should also be noted that adjusting the algorithms to a higher resolution method is not complicated. If the data were to become more precise in the future, then this adjustment can still be made. So the current method uses a single performance value for each panel.

5.1.3. The performance of a PV system

A PV system generally consists of more than one panel. These panels are then connected together in arrays. The panels can be connected in series or in parallel. It matters a great deal how these connections are made. The impact on the combined IV curve can be seen in figure 5.8.

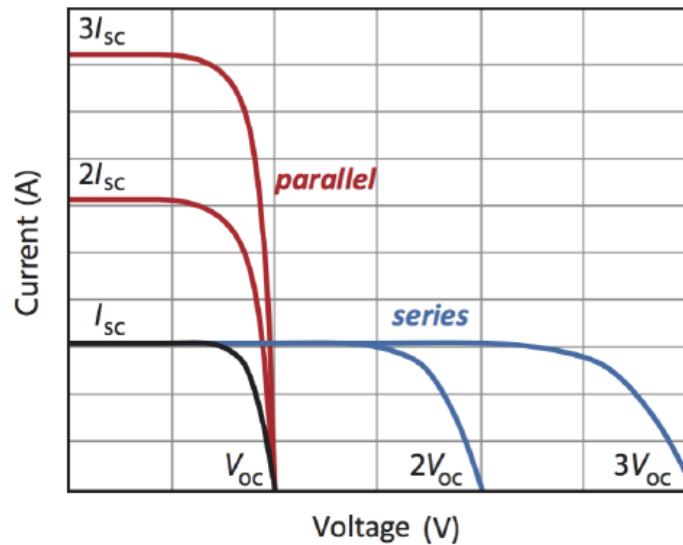


Figure 5.8: The effect of series and parallel connections on the total IV curves. Figure adapted from [8].

When panels are connected in series, their voltage is added and the current is limited by the panel producing the lowest current. This aspect is the same as with the solar cells inside the panels, as was discussed earlier. A shaded panel in a string of panels in series has the same detrimental effect as a shaded cell in a string of cells in series.

The performance of the individual panels will determine the total performance of the PV system. The total performance of a PV system with multiple panels is determined by the configuration of the BoS. The choice for the inverter and how the panels are connected is very important. In this thesis a simple calculation is used for the series and parallel connections. In a series connection, the worst performing panel is used as the performance for all panels. A possible bypass is not taken into account in this way. The performance for a string of N panels in series is then given by

$$P_{series} = N \cdot \min(P_1, P_2, \dots, P_N). \quad (5.1)$$

The performance of a parallel connection is given by sum of all the individual performances. The performance for a string of N panels in parallel is then given by

$$P_{parallel} = \sum_{n=1}^N P_N. \quad (5.2)$$

5.2. Brute force methods

The first attempt in placing the panels consisted of a more or less brute force method. The best location for a panel is a position in which there is no shading. Placing the desired amount of panels one by one ensures that every panel is in the most optimal position. The result is however that the panels are all clustered in a good area on the roof, but that they are disconnected. With several good areas on the roof, the panels might even be scattered far away from each other. An example of a clustered, but disconnected group of panels can be seen in figure 5.9.

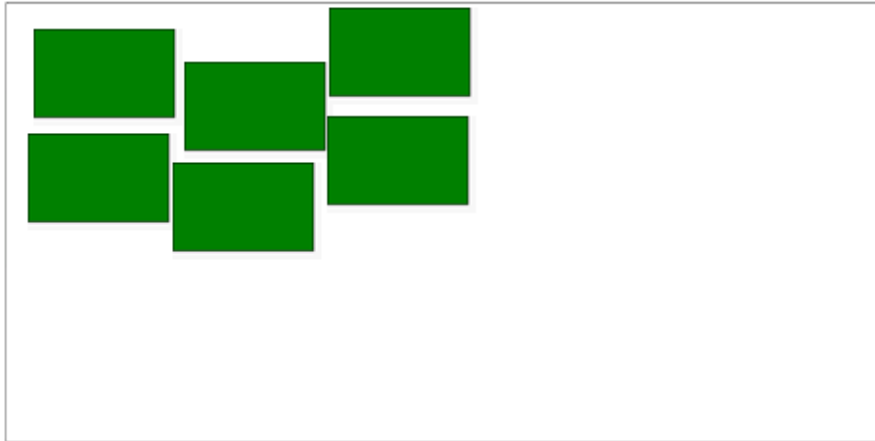


Figure 5.9: A clustered, but disconnected group of panels.

Such a layout creates roof racking problems, wiring problems and it is also aesthetically unattractive. The panels should not just be placed at the optimal location, but extra constraints should be included. Such a constraint is that the panels should be connected with each other and be aligned, either horizontally or vertically.

Another approach is when placing the panels after each other, the next panel has to be connected to the previous panels in the way described by the constraints. It can be placed above, below, left and right of other panels. It can however be the case that the first panel is placed near the edge of the roof or in between obstacles and subsequent panels won't fit. Another downside to this approach is that the other panels can end up in a very bad position, due to the position of the first panel. The method of placing individual panels was abandoned and a method involving panel sections was introduced.

Shifting a panel section

A different method involves not the placement of panels, but the placement of panel sections. The sections are all rectangular panel sections. There are however many more possible ways of making a section of panels. A few of those can be seen in figure 5.10.

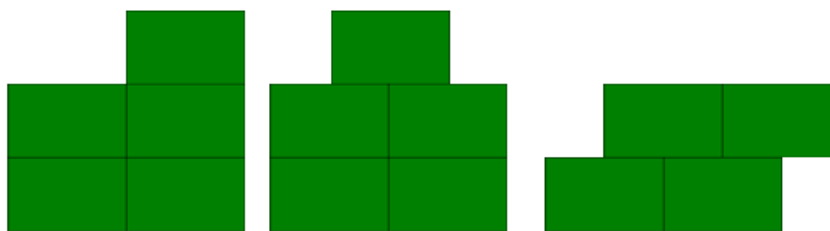


Figure 5.10: Examples of non rectangular panel sections.

These non rectangular sections are however not considered. The rectangular panel sections can be seen as a grid and the notation of a panel section is: (number of rows x number of columns). This means that a panel section of 2x3 consists of six panels with two horizontal rows of three panels on top of each other.

Finding the best location for a certain panel section can be done by shifting the section across the roof and testing all possible positions. The performance of the panel section is calculated each time the grid is shifted. The image in figure 5.11 illustrates the concept of shifting a panel section.

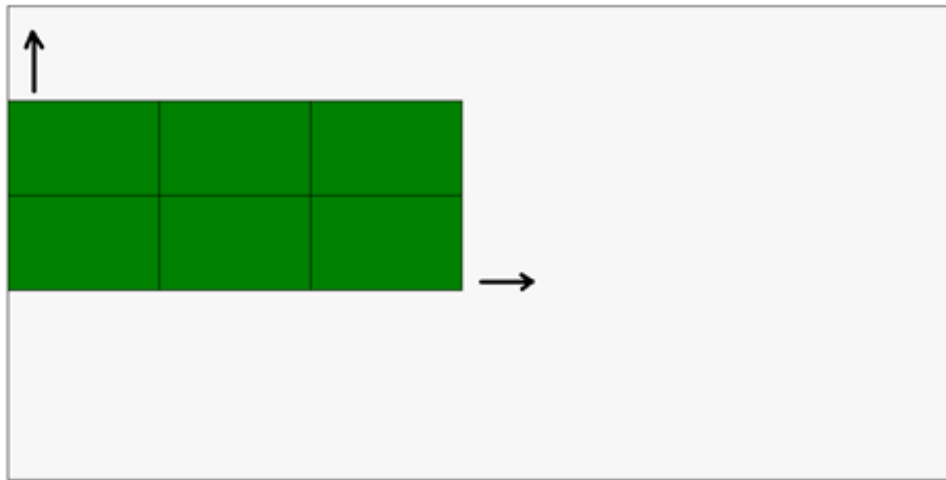


Figure 5.11: Illustration of shifting a 2x3 panel section across a roof surface.

The position that results in the best performance can easily be determined with this method. The amount of calculations required for this method depends on the size of the roof and the step size that is used for shifting the panel section.

Sometimes it is however better to use more than one panel section. The reason for this can be that a single large section doesn't fit or that the overall performance is better if separate sections are used. If there is a large shadow in the center of the roof, a panel section can be broken up into smaller pieces to be placed around the shaded area. It would be possible to check all combinations of ways of placing two panel sections on the roof and pick the best performing scenario. An issue with that method is that it is necessary to know which panel sections have to be used. This is however not known beforehand and therefore all options would have to be tried. For a single section of panels there are multiple options of how to arrange the panels. A panel section of six panels can in theory be placed in eight different ways. A function was written to determine all the possible panel section combinations for a certain amount of panels, up to two sections. A graph showing the amount of combinations can be seen in figure 5.12.

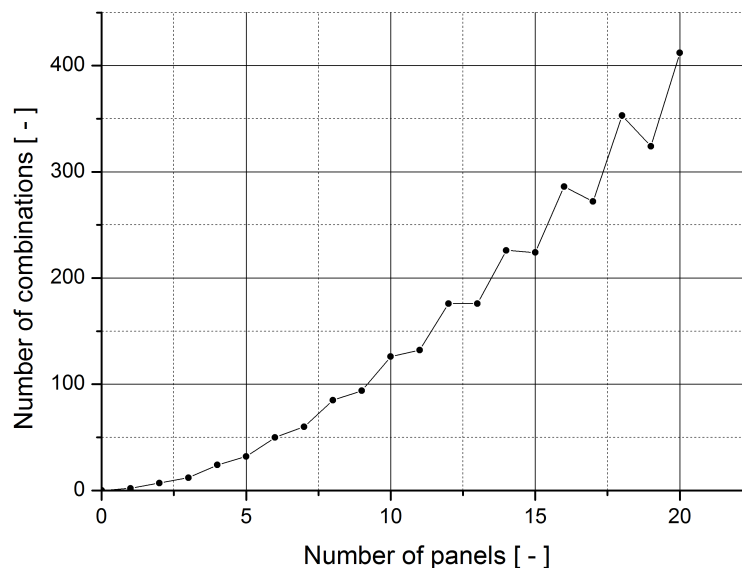


Figure 5.12: The amount of panel section combinations, up to two sections.

Going through all these possible combinations is not practical to do. These combinations are also only for combinations that consist up to two sections, more sections would also be possible in reality. For each combination every possible arrangement on the roof also has to be checked. Most of the panel section combinations are however not practical in reality. Even if the amount of combinations is reduced it will still be quite a lot of combinations. At this point the decision was made to look into a smarter algorithm. An algorithm that can deal with a problem that has a lot of possible solutions, without having to go through all the possible options.

5.3. Genetic algorithm

After the decision was made to try another approach, several different people were contacted for ideas and suggestions. One of the first ideas was to perhaps look into pattern recognition and machine learning to look for ways to utilize the information about the roofs. This information is the performance grid and the shape of the roof. Conversations with dr. David Tax and dr. Marco Loog from the Pattern Recognition Laboratory at TU Delft gave some more insights into this possibility. It was however concluded that this problem couldn't really be solved with their area of expertise. The problem was recognized as an optimization problem.

The problem can be classified as an optimization problem with several objectives, such as layout and performance. It was suggested to look into genetic algorithms. Those algorithms could be used to deal with these kinds of optimization problems. After these two meetings there were two more meetings. The first was with dr. Karen Aardal from the Applied Mathematics department from TU Delft. The second one was with dr. Peter Bosman from the Center for Mathematics and Computer Science in Amsterdam. These meetings gave insights and ideas about the development of the genetic algorithm. The genetic algorithm had some initially promising results, but eventually the decision was however made to continue with another approach. This new approach was however inspired by some of the principles of the genetic algorithm. In this section only the concept of the genetic algorithm will be mentioned and the results and ideas that were taken from it are highlighted. More information about the genetic algorithm that was developed and tested can be found in Appendix A.

General principle of the genetic Algorithm

A genetic algorithm is a method to find possible solutions to an optimization problem that has many parameters. The first genetic algorithms were introduced in the early 1970s by John Holland [22]. A genetic algorithm is a search that uses selection and variation. Several methods are described in 'Local Search in Combinatorial Optimization' by Emile Aarts and Jan Karel Lenstra [23]. This book was used for strategies and ideas during the development of the genetic algorithm. In a genetic algorithm an initial collection of pseudo random solutions to the problem is manipulated in several rounds. Each round it creates a new collection of solutions called a generation. With each generation the value of the solutions increases and this is continued until a certain stopping criterion is met.

Performance of the developed algorithm

The algorithm revolves around the idea that a good solution will be formed if a combination of panel sections can be found that matches the desired amount of panels, is well positioned on the roof and has an aesthetically appealing layout. The developed genetic algorithm was tested for several different scenarios. These scenarios were handmade roof polygons with a self made performance grid. The algorithm started to look promising, as the solutions consisted of panel sections that were placed on positions on the roof with the least amount of shading and the better looking solutions floated to the top after a layout grading was introduced. There were however several downsides to this method, which lead to the development of a new method. A concern was the fact that this method was difficult to expand to solutions with more than two panel sections. Another reason is that other methods were thought of that could accomplish the same thing in a more efficient way. It was therefore decided to choose a different approach, but based on some of the concepts of the genetic algorithm.

Concepts and ideas used in new approach

It was realized that what the genetic algorithm was doing could be accomplished in another way. The algorithm was essentially trying to match panel sections in good positions on the roof with each other to form a good overall solution that fits the amount of panels required. These good positions are called local optima. The procedure of how this was done can be read in Appendix A. This same idea is used in a new algorithm, but without the randomness factor that was part of the genetic algorithm. Another concept taken from the genetic algorithm was the idea of optimizing for multiple objectives. A grading system for the layout was developed and all of the solutions were sorted based on multiple objectives. This is also done in the new algorithm and will be explained in more detail in the next section.

5.4. Local optima search and combine algorithm

In this method the idea is to find good positions for different panel sections on the roof and then to combine these panel sections to form solutions. This is similar to the genetic algorithm, but it does it in a different way. The algorithm consists of four main stages and the search for solutions is done in three steps. All of these stages and steps will be discussed and to start an overview of the algorithm is given in the flowchart in figure 5.13.

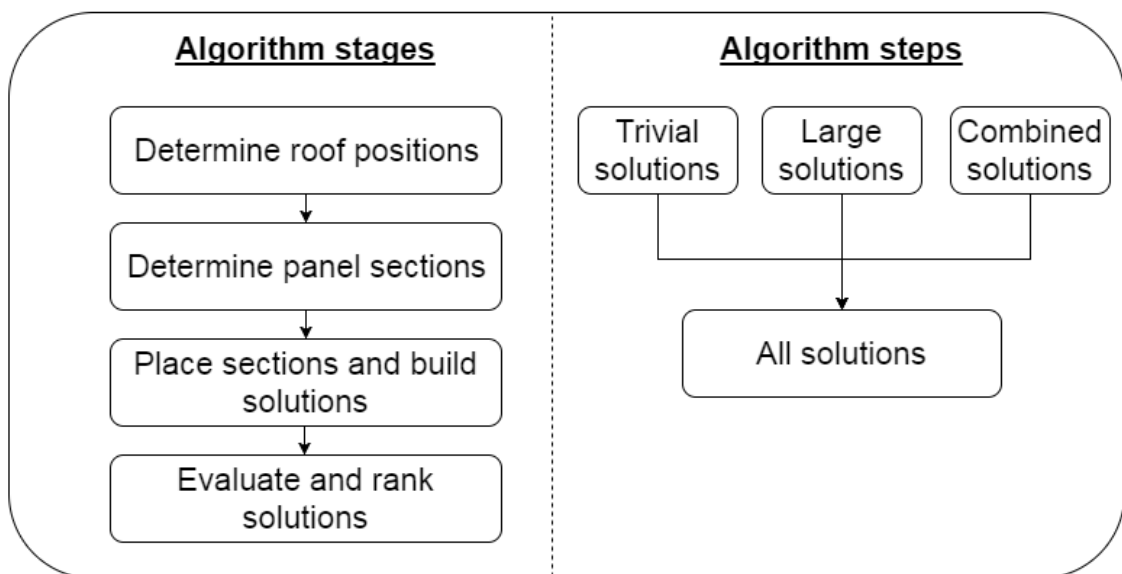


Figure 5.13: A flowchart of the stages and the steps of the local optima search and combine algorithm.

The search for solutions is split into three steps. This means that each stage is basically done three times, one time for each step. The first step looks for what are called trivial solutions. These trivial solutions consist of single panel section solutions that have a rectangular shape. These are sections that consist of all the panels that are required. It was found that quite often this is just the desired layout. The second step looks for the what is called large solutions. These are sections that are about the size of the bounding box of the roof polygon. This was implemented, because it was realized that often the desired amount of panels is almost covering the entire roof. This means that a panel section that covers the roof can be suitable as a solution if it has the right amount of panels. Roofs with obstacles will cause certain panels to be removed due to overlap. This means that a suitable solution can arise when a such a large section is used and certain panels are removed. It is also a very quick search and therefore it was added. Using this method can often also result in solutions with too many panels. These solutions are discarded for now, but excess panels could in the future also be removed. The third step consists of looking for solutions consisting of two sections. These sections are small, because they have to combine to form the required amount of panels.

Determining roof positions

An algorithm makes a list of positions across the roof. The exact locations of these positions can be adjusted. A grid of points is placed on the roof and there are several parameters that determine how this grid looks like. These parameters are the distance between the grid points, called the step size, and the distance from the edge of the roof, called the buffer.

Initially the idea was to use only the best starting positions, based on the shading on the roof. A grid of points is placed on the roof and the level of shading is determined in each point and then only the best points are saved. An example of this concept with 20 points saved can be seen in figure 5.14.

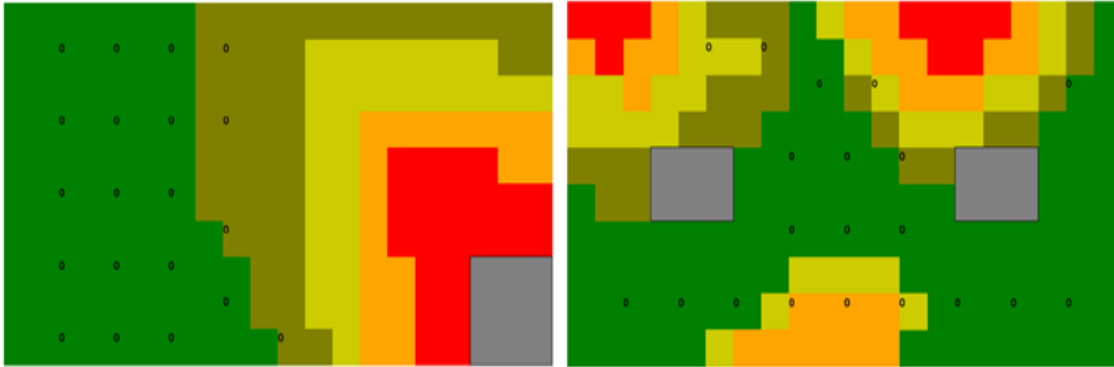


Figure 5.14: Points saved after a probing algorithm is applied. Green areas receive the least amount of shading

The benefit of this method is that only good areas of the roof are used to find solutions and this reduces the amount of computations done. This approach was later abandoned for two reasons. First of all the algorithm uses less of the available roof to find solutions, which could mean that a solution is not found at all. Secondly, the differences in performance across the roof wasn't as high in reality as initially was expected. Unless major obstacles are present, the differences aren't very big. This means that by choosing just the best points, potentially good positions are thrown away. So instead of selecting only the best positions, all positions are used that don't overlap with obstacles and fit on the roof. The roof positions are used to place panel sections. These panel sections will then later be used to build a solution.

Each step of the algorithm uses different roof positions. For the trivial solutions, the buffer is chosen such that at least every trivial panel section will fit on the roof. For the large solutions only roof positions near the center of the roof are used. The combined solutions require the most roof positions and has a buffer of only half the width of a panel. The step size for the position grids for all these steps can be varied.

The algorithm for getting the roof positions is given in pseudo code by algorithm 3 in Appendix B.

Determine panel sections

To make sure that the right solutions can be build, several different panel sections have to be placed at each roof position. A system with ten panels can be build with a single section of ten panels or with smaller sections that together have ten panels. A list panel sections is placed at each roof position. A selection of panel sections and a roof with roof position is shown in figure 5.15 in order to illustrate this concept.

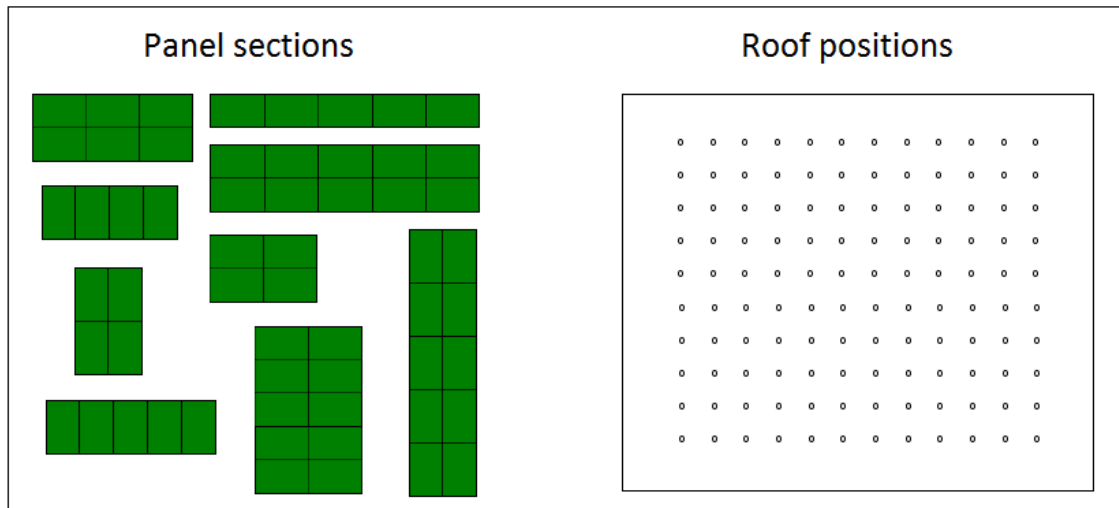


Figure 5.15: Panel sections to be placed and roof positions.

The list of panel sections to use is a very important parameter for the entire algorithm. More panel sections will require more computation time, but fewer sections can result in not finding a solution.

The list of panel sections is based on the roof surface and on the design preferences. The first thing that is considered is the design preference, which consists of a range of desired amount of panels. For example a design preference could be a PV system of 10 to 12 panels. The algorithm determines how a PV system could be built that consists of this amount of panels and could contain up to two panel sections. In the future, solutions for more than two panel sections can also be included. This was not done, because looking for combinations with more sections increases the computation time a lot and almost no PV systems that were seen during the development consisted of more than two sections anyway. A roof that contains physical obstacles will result in a different list than a roof without physical obstacles. The reason for this is that overlap with obstacles results in the removal of panels, which can lead to the situation that not enough panels are left in the solutions. For example, if ten panels are desired and a 2x5 section has one panel removed it now has too few panels to become a possible solution. A panel section of 2x6 could have 2 panels removed and still be left with a possible solution of ten panels. How much larger these sections should be and which ones to use is a parameter setting that can be adjusted.

The algorithm to determine the panel sections is given in pseudo code by algorithm 4 in Appendix B.

Place sections and build solutions

Each panel section will be placed on each starting position in both landscape and portrait orientation. This results in a big collection of all panel sections placed on all starting positions. The performances of these panel sections at these positions are then calculated. This performance is for a series connection. The next step in the algorithm is to build solutions with the list of panel sections. This is done by combining the panel sections to form a solution that fits the amount of panels desired. The performances of the combined sections is the sum of the performances of the single sections. This is however not possible if the two panel sections overlap with each other. Then some panels are removed and then the question becomes which section has to remove those panels. The section values then also have to be evaluated again. There are many possible solutions in which the sections overlap and doing these extra computations took a lot of time. It was therefore decided to discard the solution if the two panel sections overlap with each other.

Evaluate and rank solutions

The final stage of the algorithm is to evaluate the solutions that were formed. The evaluation is based on the performance and the layout. The evaluation of a solution is a list that contains five values. The

first factor is the total performance and the second factor is the average performance per panel. The other three factors in the evaluation are based on the layout. The total score of a solution is called the fitness (F), which is terminology used in genetic algorithms. This fitness is given by $F = [P_{total}, P_{avg}, S, H, L]$. These last three factors are explained in chapter 6, as well as the method of ranking the solutions.

The algorithm also produces duplicate solutions that have to be filtered out. This happens, because different panel sections can have panels removed in such a way that they end up being the same. This concept is illustrated in figure 5.16.

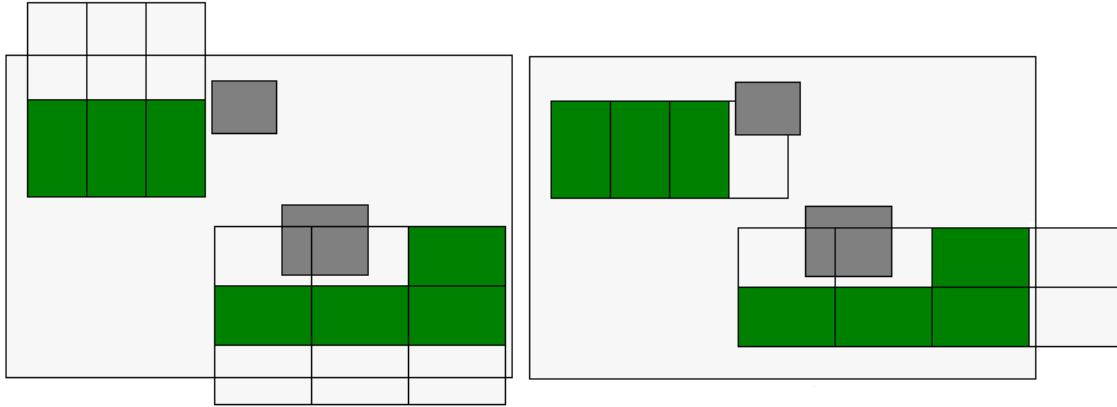


Figure 5.16: Illustration of how duplicate solutions can arise.

In order to distinguish these duplicates from each other an extra notation was developed. This notation is a matrix that has the size of the panel section. For each element in the matrix there is a 1 for a landscape panel, a 2 for a portrait panel and a 0 if there is no panel. The matrix notation of panel section changes when it has panels that overlap with obstacles.

The matrix notation is used to determine whether solutions are duplicates of each other. The outside rows and columns that contain zeros are removed. This reduces the matrix to a smaller version and will reduce matrices of the same layout to the same matrix. The matrix notations of the panel sections in the left scenario of figure 5.16 and their reductions are:

$$\begin{pmatrix} 0 & 0 & 0 \\ 2 & 2 & 2 \end{pmatrix} \rightarrow (2 \ 2 \ 2),$$

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

The matrix notations of the panel sections in the right scenario and their reductions are:

$$(2 \ 2 \ 2 \ 0) \rightarrow (2 \ 2 \ 2),$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

A duplication is found if two solutions consist of the same reduced matrices at the same positions on the roof. All of the duplicates are removed from the collection of solutions in this manner. The algorithm presents the best solutions without duplicates, that fit the system requirement and that have been sorted according to their layout. This method is tested and the results of these tests can be read in chapter 9.

Flat roof implementation

On a flat roof the row distance has to be preserved, which means that creating panel sections has to be done differently. If multiple panel sections are combined, this row distance also has to be preserved. One way of doing this is to add an edge to the panels when placing them on the roof. This will make sure that the row distance is always ensured. This concept is illustrated in figure 5.17.

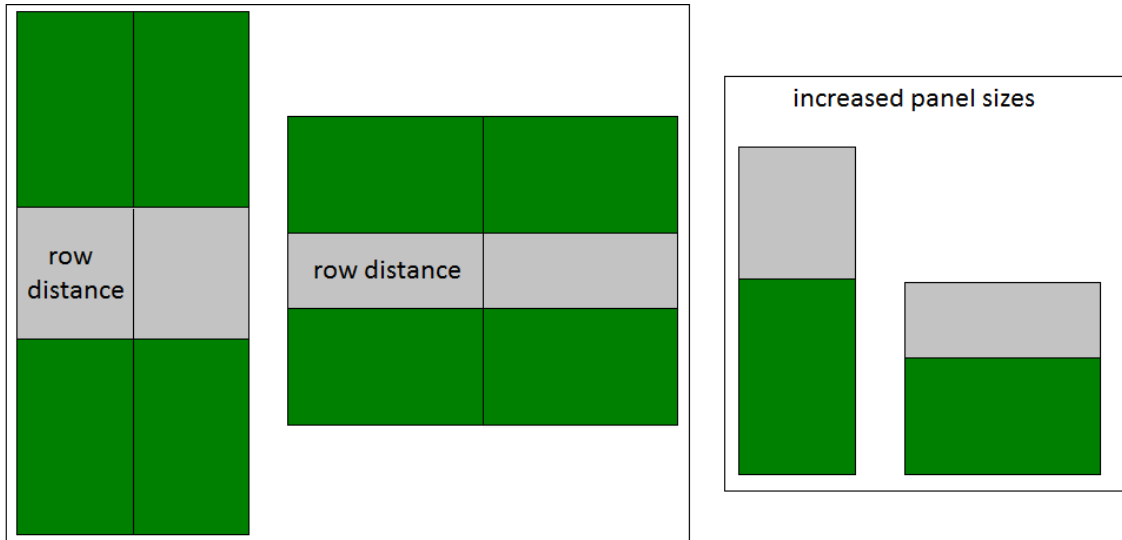


Figure 5.17: Adding an extra edge to the panels will ensure the row distance.

The size of the edge added for landscape panels is not the same as for portrait panels. When determining the overlap with obstacles on the roof the actual panel size has to be used. Otherwise the algorithm will check for overlap and return false positives. An example of such a false positive is shown in figure 5.18.

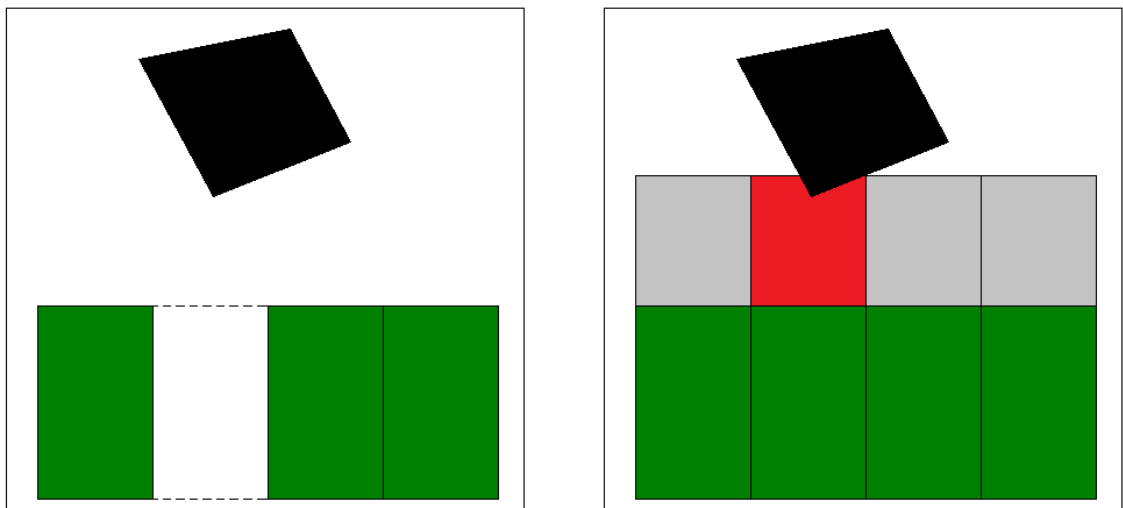


Figure 5.18: The extra edge on panels can cause a false positive when checking for obstacle overlap.

This means that the sizes of the panels used in the algorithm have to be carefully managed. When placing the panels the original size is used to check for overlap with obstacles. When combining different sections, the enlarged sections have to be used to check the overlap with the other section.

The rest of the algorithm works the same, except for the layout grading. The layout is usually of less importance on a flat roof, because it is less visible. There can also be other factors that play a

role. This means that a different layout grading system has to be used. This was however not yet investigated or developed. The same layout grading as for pitched roofs was used for now.

The algorithm for the finite panel placement algorithm is given in pseudo code by algorithm 5 in Appendix B.

6

PV system layout grading

The finite panel placement algorithm will present many different solutions that fit the design requirements. There are however many solutions with a very similar performance, but with different layouts. In order to make a decision about which layout is best, a grading system is developed. The information used for this part of the research is obtained by talking to some PV installers who came by at Solar Monkey. They had experience with installing PV systems and with making decisions about the layout of the system. The aspects of the layout are divided into three categories, these categories are: performance aspects, installment aspects and aesthetic aspects. All of these aspects are discussed in this chapter and how they are incorporated into a grading system.

6.1. Performance aspects

The performance of the system is dependent on where the panels are placed on the roof. This means that the performance of a PV system can also be linked to the layout of the system. Sometimes a certain performance can only be achieved with a certain layout.

Another performance aspect is the wiring in a PV system. The layout of a PV system determines to a certain extent the length of the wires that are needed to connect the entire system. Cable losses, depend on the length of the wires. This means that certain layouts may result in more cable losses than other layouts. These losses are however only a few percent of the total performance for residential houses [12] and optimizing this will only result in a small performance benefit. Perhaps including this factor can be significant for very large systems, but for now this aspect is not included in the layout grading.

6.2. Installment aspects

The layout of a PV system determines how it has to be installed. Some layouts are significantly easier to install than others. The material requirements also differ for different layouts. There are aspects of the layout that have such an impact on the installment that they require attention in the designing process. Several relevant installment aspects are discussed in this section.

6.2.1. Number of panel sections

The algorithm combines different panel sections to form a solution. This means that a layout can consist of one or multiple different panel sections. For a PV system of only eight panels there are already many options, a few of those are illustrated in figure 6.1.

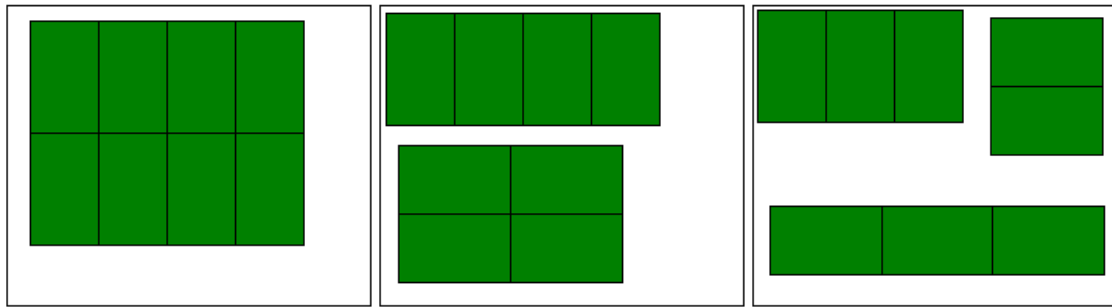


Figure 6.1: Layouts with different number of sections for a PV system with eight panels.

Using multiple panel sections can ensure a better use of the good positions on the roof. This will lead to a higher total performance. It is however easier to mount all the panels in a single section. In terms of the installation it would therefore be preferred to have the least amount of panel sections. The number of panel sections is one of the layout factors and is called S .

6.2.2. Vertical or horizontal placement

Sometimes there is also the decision whether to place a panel section in a vertical or horizontal orientation. An example of this can be seen in figure 6.2.

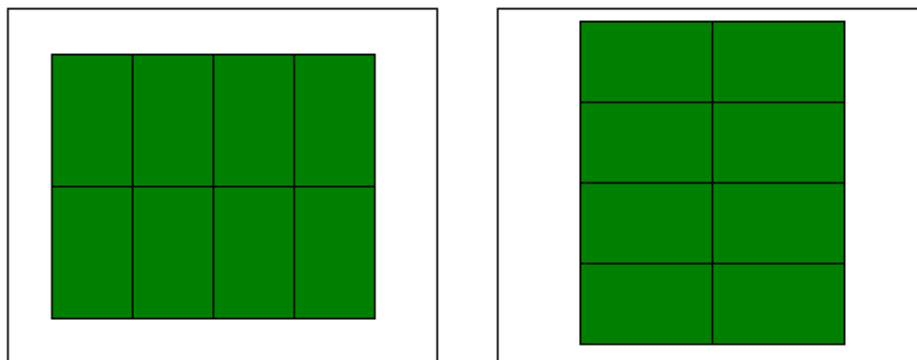


Figure 6.2: Horizontal versus vertical placed panel sections.

According to the installers a panel section that is mounted vertically is more difficult to install. The mechanic has to move across the roof vertically, instead of moving horizontally. This vertical movement is more difficult and more dangerous. Therefore a horizontal placement for the section is chosen if that choice arises. This was however not yet included in the algorithm.

6.2.3. Compactness of a panel section

There are many ways of placing a certain amount of panels in a single section, especially when there are obstacles on the roof. A method for classifying these different layouts is by determining how compact the layout is. A measure of compactness of a panel section is defined for the layout grading. An example of two panel sections with a different compactness can be seen in figure 6.3.

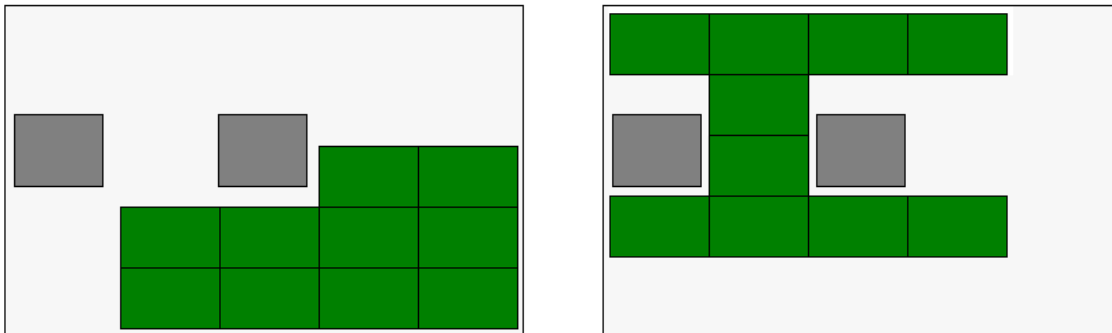


Figure 6.3: Two different panel sections with 10 panels with different compactness.

The layout on the left is seen as more compact than the layout on the right. Compact layouts are easier to install and generally look better and will therefore be a preferred option. The convex hull is used as a measure for how compact a layout is. The convex hull of a section is section that can be seen as the original panel section, but with a tight elastic band around it. For three different panel layouts the convex hull can be seen in figure 6.4.

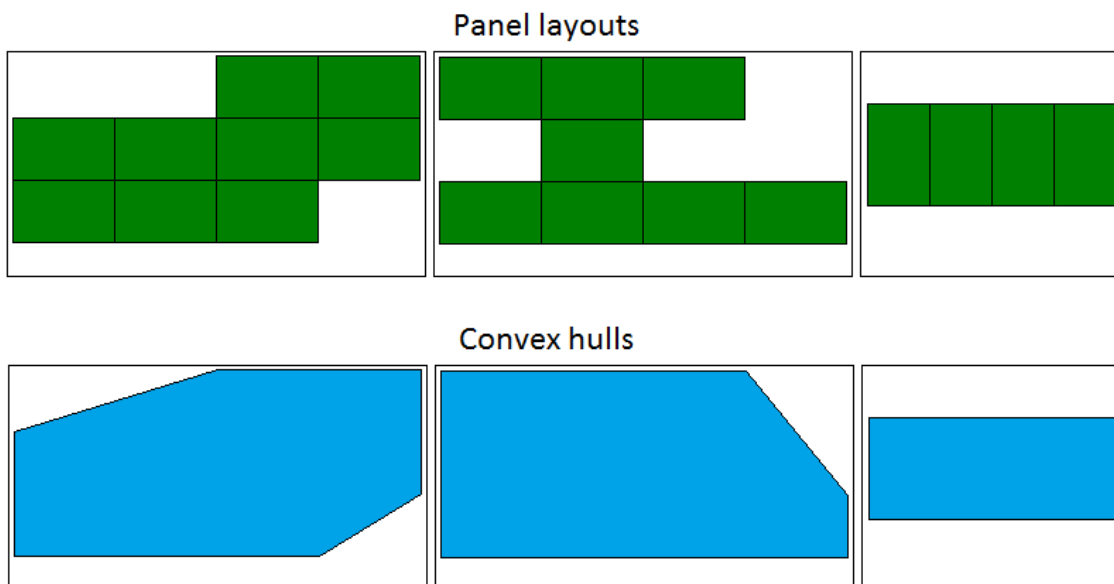


Figure 6.4: The convex hull for three different panel layouts.

A score based on the convex hull is given by determining the ratio of the surface area of the original panel section and that of the convex hull. This is called the hull factor (H) and is given by:

$$H = \frac{A_{panels}}{A_{hull}} \tag{6.1}$$

For a perfectly compact system the hull factor is 1. The hull factors of the layouts in figure 6.4 are from left to right: 0.86, 0.73 and 1. This hull factor is maximized for the best layout.

6.2.4. Mounting material

The panels are mounted on a rails that is attached to the roof. A simplified view of such rails behind the panels can be seen in figure 6.5.

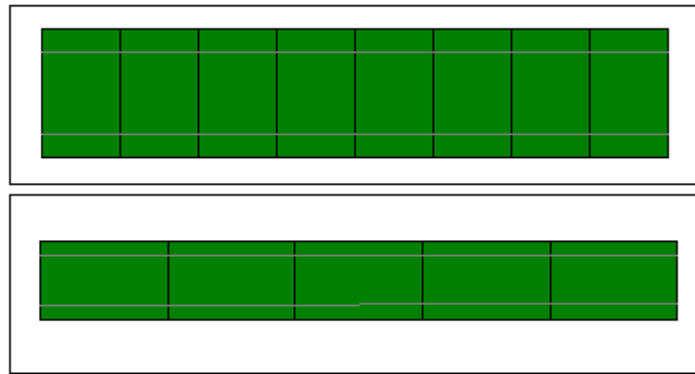


Figure 6.5: Mounting rails behind the panels.

About the same amount of rails is required for the portrait and the landscape row. There are however eight portrait panels in the row and only five landscape panels. This shows that placing the panels in portrait orientation will result in less material requirement. The installers said that a row of panels placed in portrait orientation is cheaper to place than a row in landscape orientation. This is directly linked to the smaller amount of rail material that is needed. The third layout factor that is defined is the rail length L . The optimization is to minimize L .

6.3. Aesthetic aspects

An important aspect when it comes to the layout of the PV system is aesthetics. This was stressed by the installers that were talked to. Two aspects about the aesthetics is discussed in this section, the customer preference and the impact the aesthetics have on public opinion about solar PV.

Customer preferences

A research by Milieucentraal under potential customers of PV panels showed that about 8% of respondents don't buy solar panels, because they find them to be ugly [24]. This means that there are people that do see the energy and financial benefits, but are put off by the aesthetic aspects. The installers said that their customers like to have a layout that looks appealing, even if that layout may not result in the highest performance. This means that optimizing for the most optimal performance regardless of the layout is not a good option.

There is also an additional factor that plays a role. Placing solar panels on your roof will generally increase the value of your house, which can be interesting if you are interested in selling it. A very unattractive looking PV system can however have negative impact on potential buyers, it can make the house harder to sell.

In the current algorithm the aesthetics are taken into account by going for the most compact layouts and the ones with the least amount of sections. Further research into the actual preferences of customers could add more factors to the aesthetic gradings of the solutions.

Public opinion considerations

As was mentioned earlier, quite some people still find the aesthetic aspects the largest barrier for buying solar panels. Seeing unattractive looking PV systems on other houses might even increase this barrier. This means that even if some extra performance is gained for a single house with bad aesthetics, that might be bad for the market as a whole. It is very difficult to consider public opinion in this algorithm, but it does emphasize the importance of aesthetics.

6.4. The grading system

Three factors have been defined, namely S , H and L . Together with the performance values they form the fitness $F = [P_{total}, P_{avg}, S, H, L]$. Each of these elements in the fitness is an objective that has to be optimized. In order to determine which solutions are better, all of these elements have to be considered.

Ranking the solutions is done based on multi objective optimization. Determining which solutions are better than others is often a subjective matter, it depends on which objective is valued the most. There are however some solutions that are objectively better than others and it is helpful to identify which ones those are. When a certain solution is objectively better than another solution, it dominates that other solution. This is the case if it has a higher score for at least one objective and all the other objectives are equal or better than the other.

An algorithm is used that sorts the solutions based this concept. It is called A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization (NSGA-II) [25]. The solutions that are not dominated by any other solutions form rank 1 and these solutions form the Pareto front. The next rank will be a group of solutions that is only dominated by rank 1 and is called rank 2. This process of ranking continues until all solutions are placed in a rank. For the factors L and S this can be visualized in a graph. The L factor is placed on the horizontal axis and the S factor on the vertical axis. Both of those parameters have to be minimized, which means that the best solutions placed in this graph should be at the bottom left. An image of solutions placed in this graph can be seen in figure 6.6.

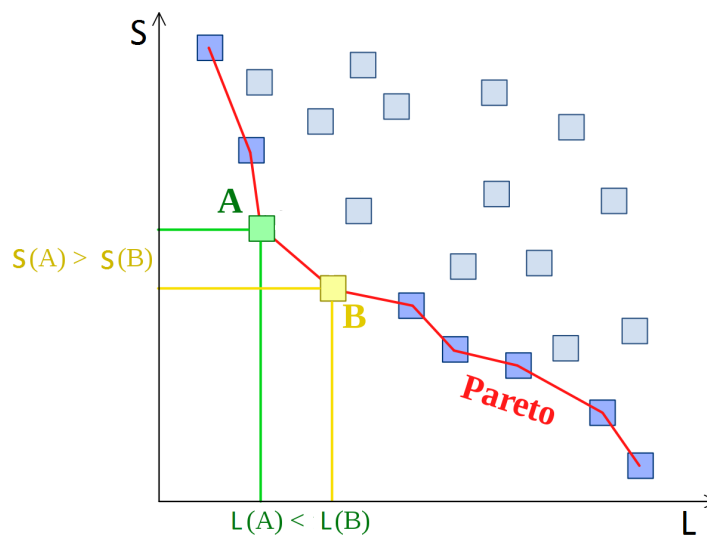


Figure 6.6: Solutions placed in a graph with the factors L and S on the axes. Figure is adapted from [26]

All of the solutions in the Pareto front are not dominated by any other solutions. A solution that is not in the Pareto front is not necessarily dominated by a solution that is in the Pareto front. Sorting the solutions like this will however reduce the possible suitable solutions drastically and fast. The first few ranks could contain the preferred solutions.

After this sorting has taken place the algorithm will present the top results. Then a subjective grading has to take place. Different solutions will be in the same rank and then a choice has to be made about which objective is viewed to be more important. One solution might have a slightly better performance, but looks less attractive. It is then up to the person that gets the design to make a decision.

7

Inverter optimization options

The type of inverter to use is an important decision when designing a PV system. Not all inverters are suitable for every panel layout. In this chapter several different inverter types are discussed, as well as the methods that were developed to determine the optimal inverter for a certain PV system. This chapter discusses ideas about inverter sizing, inverter choice and string configurations that leads to an optimal inverter choice. The ideas are still in the development stage and have had only limited testing and confirmation. It can however be the basis of a further research in which the algorithms that were developed are optimized. They could then be implemented as addition to the panel placement algorithm for a more complete automatic PV system design algorithm.

7.1. Inverters

An inverter is required to convert the generated DC power by the solar panel into AC power that can be used for home appliances. There many different types of inverters and the optimal choice is different for each situation. The inverter does more than only converting the DC power to AC power. It also regulates the performance of the PV modules by tracking the maximum power point [27]. It adjusts the load so that the voltage is changed and the panel operates at the MPP. The inverter can only handle a certain amount of input power, which means that the size of the inverter is determined by the number of panels. Some commonly used inverter types are briefly discussed to determine which types are best in which scenario. Then several methods are discussed that determine the optimal inverter.

7.1.1. Central and string inverters

A common method is to connect all the panels to a single central inverter. An image of this can be seen in figure 7.1.

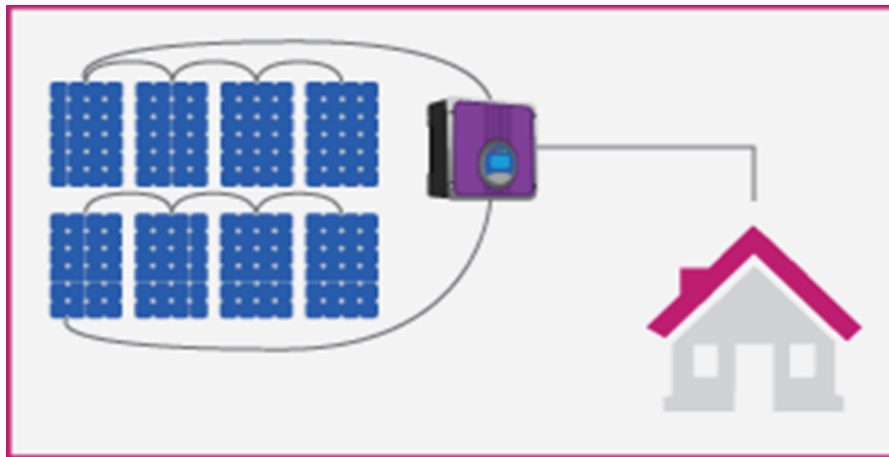


Figure 7.1: A string or central inverter PV system. Image adapted from [28]

A large PV system can have several separate strings with each string connected to a single string inverter. The maximum power point tracking is done for all the panels that are connected to the same inverter. This means that the MPPT will do the same load adjustment for all the panels. Since the modules can receive a different irradiance, this is not optimal. The panels in the string are also connected in series, which has the big shading vulnerability, as was discussed in chapter 5. These inverter setups are however the cheapest option. It is a good option if there is little to no shading on the panels in the string.

7.1.2. Micro inverters

Instead of connecting multiple panels to the same inverter, each panel can also be connected to a separate inverter. A small inverter for each panel is also possible and those inverters are called micro inverters. In this configuration each panel has its own MPPT device. This means that all the modules will be performing at their best. This means that the total performance will be better than with string or central inverters. Such a setup does not have all the panels connected in series, which means that shading isn't as much of a problem. An illustration of such a configuration can be seen in figure 7.2.

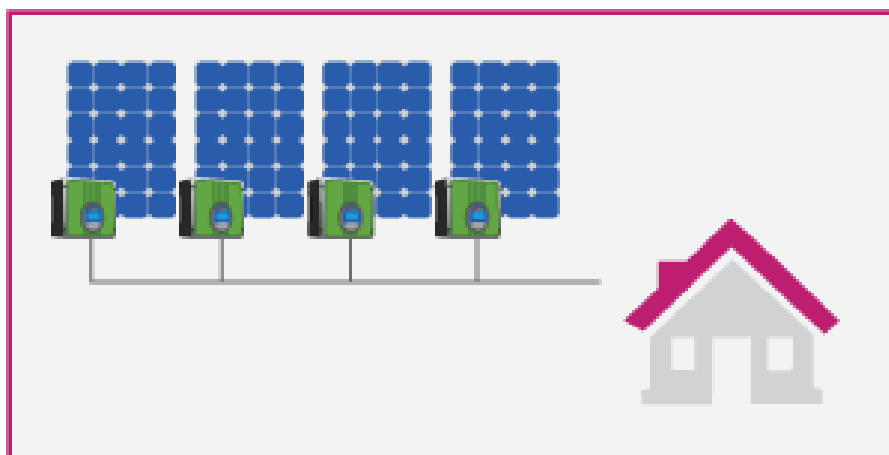


Figure 7.2: A micro inverter PV system. Image adapted from [28]

With a micro inverter configuration each panel will produce the most energy, but is also a more expensive option than a string or central inverter. If the panels receive a lot of shading or if they are mounted in different orientations or tilt angles it can be a good option. The cost of the inverters and the performance difference should be compared to make a decision between inverter types.

7.1.3. Power optimizers

The third type of inverter that is discussed uses power optimizers. This is a relatively recent development that has the independent panel benefits of micro inverters, while being a cheaper option. An illustration of a system with optimizers can be seen in figure 7.3.

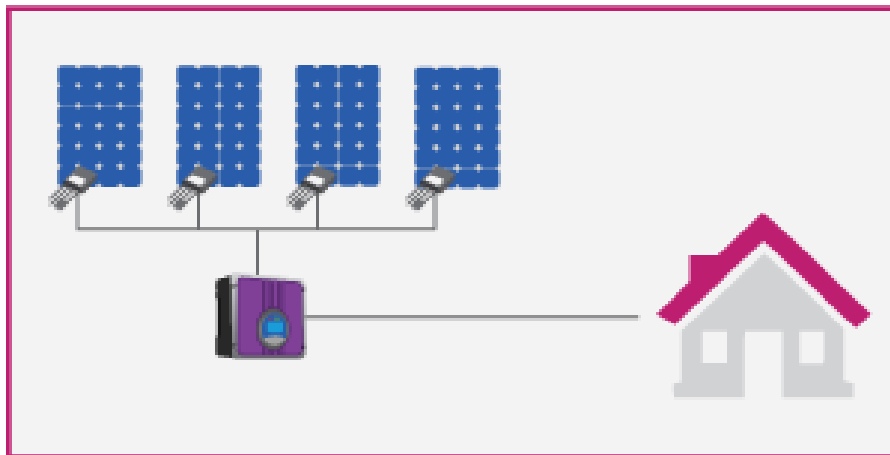


Figure 7.3: A power optimizers inverter PV system. Image adapted from [28]

The power optimizers have MPPT devices at each panel, but don't convert the DC power to AC. The panels are connected to a DC bus and that is connected to single string inverter. It has the same principle of the micro inverters, in the sense that shading of panels in the string aren't as bad as with a string inverter. The shading of the panels can determine which inverter type is best suitable for that particular situation.

7.2. Panel miss match

The shading of panels in a PV system are a very important factor in the decision of the inverter. The yearly performance of a panel does not give all the information required to make a well informed decision for an inverter. It is important to know whether the panels on the roof receive a different irradiance at the same point in time. This is called the miss match between panels. An example would be two panels with one facing east and one facing west. These panels can receive about the same total irradiance in a year, but they get that at different times of the day. In the morning the sun is in the east and the east panel will receive a lot of irradiance, but the west panel won't. In the late afternoon it is the other way around. Connecting these two panels in series means that the panels will both only produce the same as the worst panel at any given time. The actual output of the system would be much less than would be expected by only looking at the yearly performance.

Differences in irradiance at the same time can also occur due to shading. The shading caused by the clouds is nearly impossible to predict, so that won't be considered here. The partial shading that occurs due to surrounding objects is considered. Only looking at a total amount of shading won't give enough information. The way in which the shadows move across the panels is important. Figure 7.4 shows two situations in which the total irradiance received is the same, but the way in which the panels are shaded is different.

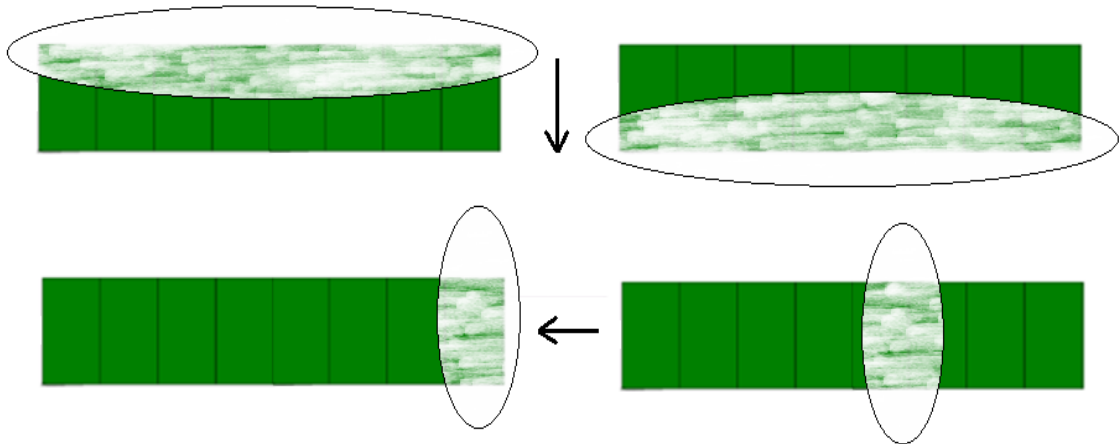


Figure 7.4: Differently moving shadows, but with the same total shading per panel.

In the situation at the top all the panels are shaded at the same time and in the situation on the bottom the panels are shaded one by one. For simplicity let's assume that each panel is shaded for a total of one second. This means that the shadow in the top scenario moves across the panels in about one second. The shadow in the bottom scenario will move across the panels in eight seconds and spends about one second above each panel. The total time that each panel is shaded is the same in both scenarios. With micro inverters there won't be much of an output difference, because each panel is treated independently. With a string inverter the scenario on the bottom will however be much worse, because the string is limited to the worst panel. That means in this case that the performance of the string will be reduced to zero for the entire eight seconds, while this is only for one second in the scenario at the top. In the next section a method is developed that predicts the mismatch between panels.

7.2.1. Relative intensity, obstacle view, and sun tracks

To determine the mismatch between certain modules a few concepts are used that are also used by Solar Monkey to predict the performance of panels. These concepts are the relative intensity, the obstacle view and sun tracks. These concepts will be briefly explained. Not how they are obtained, but what they are and how they can be used.

Relative intensity

The relative intensity is based on the orientation and the tilt angle of the panel. Diffuse light comes from all directions. Not all directions are however equally relevant for a panel. A module facing south will for example not receive any irradiance from the north. This means that north is not a relevant direction for a panel facing south. The relative intensity consists of a 1000 by 250 array that shows how much each direction contributes to the total received diffuse irradiance for a panel with a certain tilt angle, orientation and position on earth. An example of a relative intensity plot can be seen in figure 7.5.

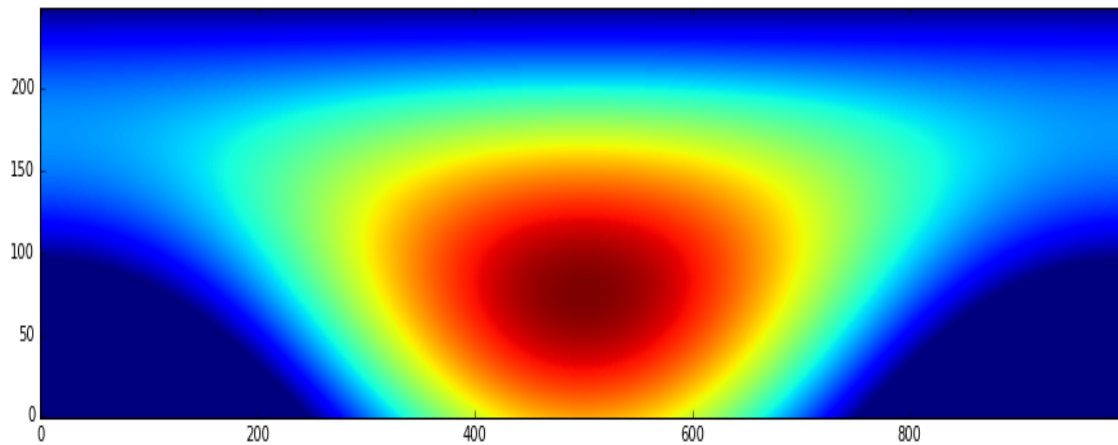


Figure 7.5: The relative intensity for a module oriented south with a tilt angle of 35 degrees.

Red means a high contribution and blue means little to no contribution. The horizontal axis represents the azimuth, split in 1000 points. South is defined as point 500. The vertical axis represents the altitude with 250 being 90 degrees.

Obstacle view

The obstacle view is created with the height data and is the 360 degree view from a certain point and contains the obstacles that can be seen from that point up to certain distance. The view is used in the shadow analysis to determine the performance of a panel. An example of such an obstacle view can be seen in figure 7.6.

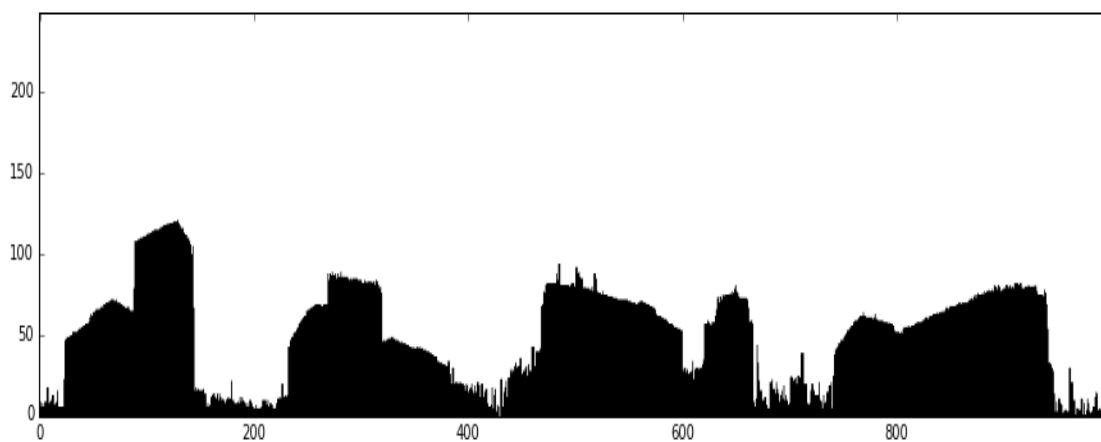


Figure 7.6: The obstacle view at a certain location.

The obstacle view indicates the directions that are blocked due to surrounding obstacles. The obstacle view is a 1000 by 250 array with ones and zeroes. A zero represents a pixel or direction that is blocked by an obstacle and is shown in black. The pixels that are not blocked are ones and are shown in white.

Sun tracks

Sun tracks are used for the contribution of direct sunlight. The direct sunlight does not come from all directions, but only from the direction directly towards the sun. The sun moves around the horizon during the day, rising in the east and setting in the west. The precise orientation of the sunrise and sunset differs from day to day. The height that the sun will rise also differs from day to day. The sun tracks depend on the location on earth. The sun tracks are a 1000 by 250 array with ones and zeroes.

The ones form the path of the sun across the horizon during the day. The daily sun tracks are bundled in monthly sun track files. The sun tracks for the months January and May can be seen in figure 7.7.

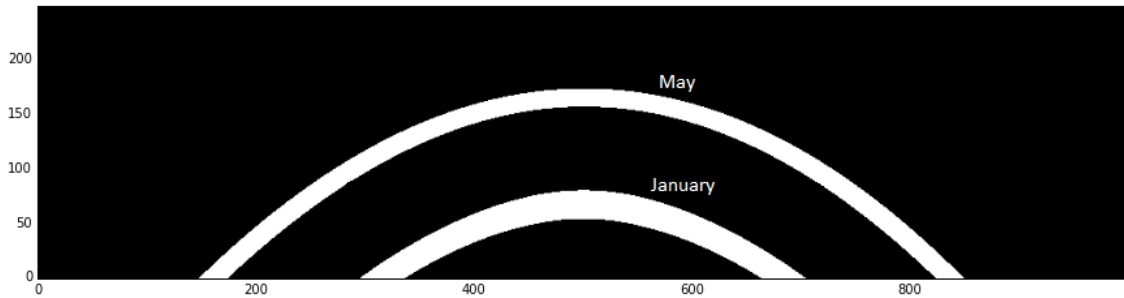


Figure 7.7: The sun tracks for the months January and May at a certain location.

These sun tracks are used to calculate the contribution of the direct sunlight on a panel. In the next section these concepts are used to develop methods to predict the miss match between panels.

7.2.2. Miss match method 1

The obstacle views are used to determine the miss match between panels. They are used to predict when a panel will be shaded. In order to use these obstacle views, a few things about how these obstacle views can be interpreted is discussed. When determining the miss match between two panels, the obstacle views are compared with each other. Some comparisons are done with a few simple artificial obstacle views. An example of two different obstacle views can be seen in figure 7.8.

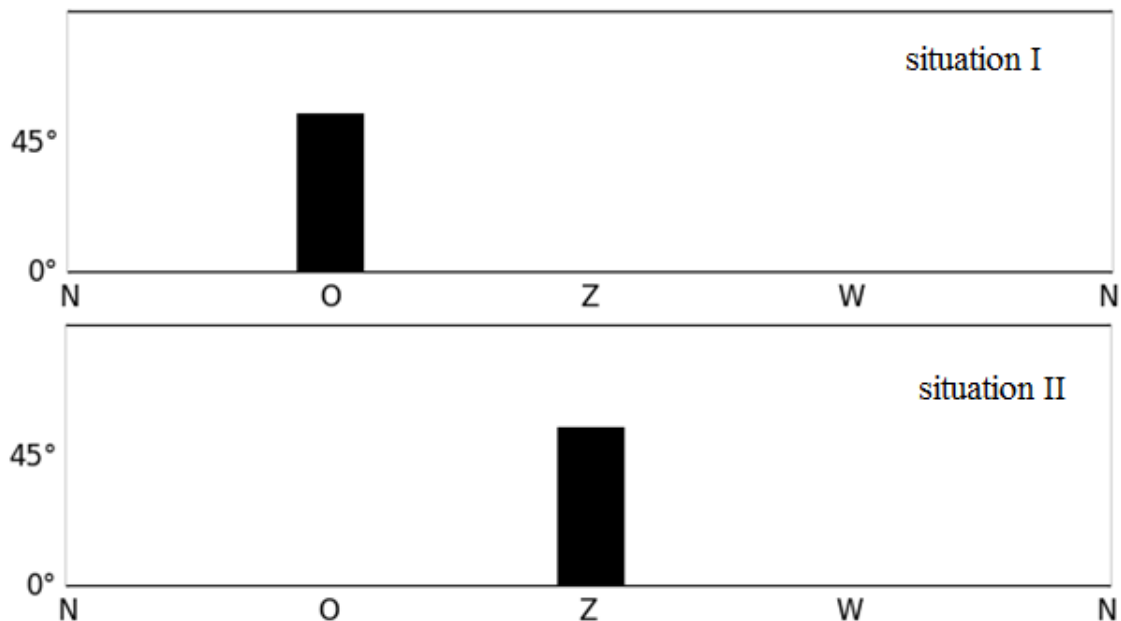


Figure 7.8: Two different artificial obstacle views.

The obstacle views show a horizontal difference, which can be interpreted as difference on a daily basis. The sun moves from east to west. In this case the sun is generally blocked in the morning for situation I and the sun is blocked at noon in situation II. This will lead to quite a significant difference in irradiance at the same point in time. This difference will be there every day. A different example of two obstacle views is not a horizontal difference, but a vertical difference. An example of this can be seen in figure 7.9.

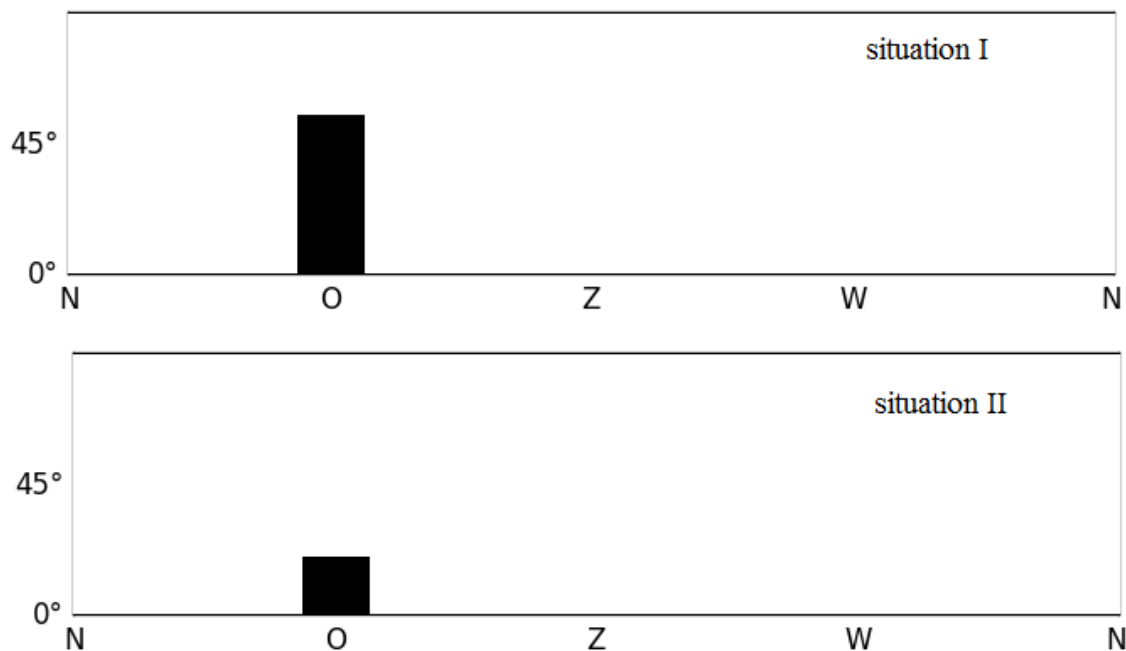


Figure 7.9: Two different artificial obstacle views.

A vertical difference is a difference on a seasonal basis. The obstacle blocks the sun somewhere in the morning every day and at the same time for both panels. It depends on the height of the obstacle in which season this difference is most significant. This difference is overall less important than horizontal differences, because the difference will not be there every day. It will only be there during seasons where the sun will rise above the lowest obstacle.

The most simple approach to define a miss match would be to just take the difference between the two obstacle views. This is however too simple, as it does matter where the differences occur. The orientation and tilt angle of the panels should also be considered. This is where the relative intensity and sun tracks play a role. The relative intensity behind the obstacles indicate how significant an obstacle is for the diffuse component. The sun tracks and relative intensity behind the obstacles indicate how significant an obstacle is for the direct sunlight component. The significance of a certain change in obstacles depends on the relative intensity and sun tracks that are behind the obstacles. Only looking at total amounts can however still lead to misleading results. An example of two obstacle view scenario's for panels facing south where this is the case is shown in figure 7.10.

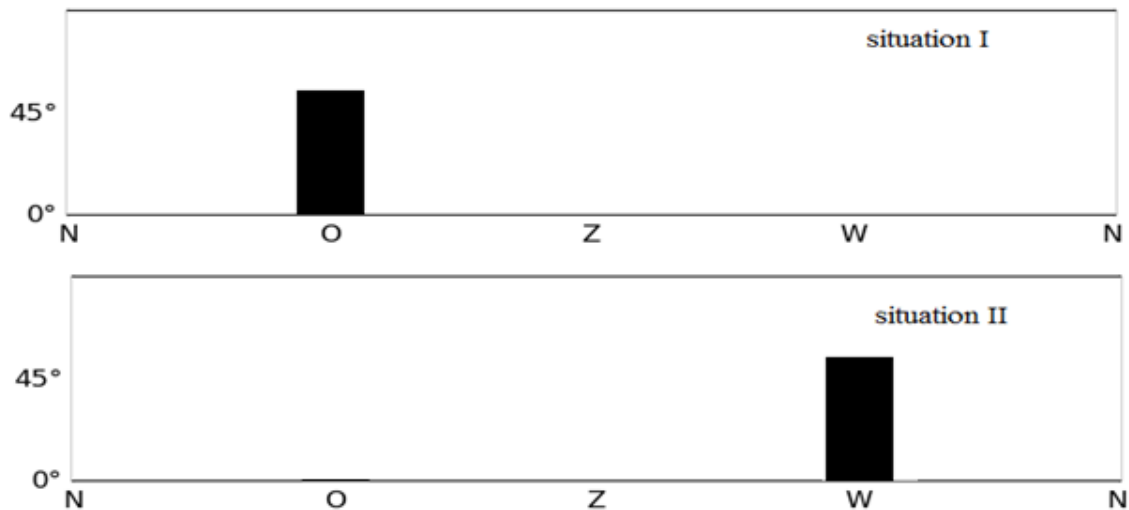


Figure 7.10: Two artificial obstacle views for panels facing south.

In this example the total relative intensity behind the obstacles can be the same. These panels will however still have a significant miss match. It is a similar story as the east versus west orientated panels with a similar total yearly performance. The method that is eventually used is based on the performance of panels in series, compared to the sum of their separate performances.

A way of determining what the yield would be for panels in series is to lay the obstacle views on top of each other. This is done, because the panels will be limited to the lowest performance. This means that the panel in situation II will also experience the obstacle of situation I, because it will also be limited to the performance of I when the sunlight is blocked by that obstacle. Combining the obstacles is shown in figure 7.11.

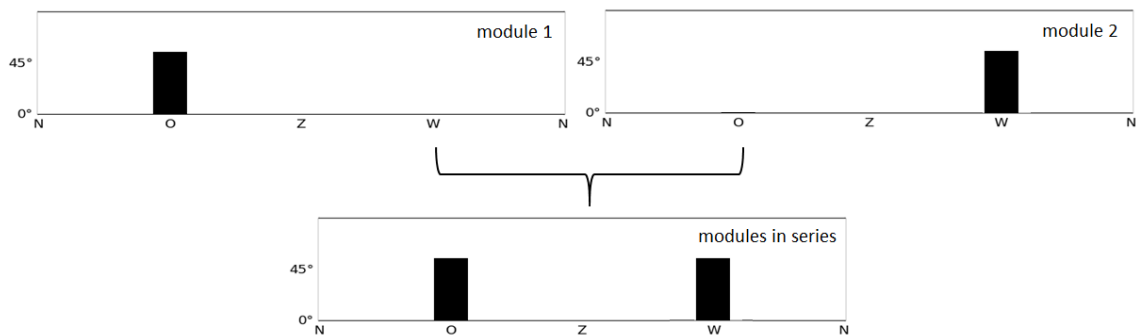


Figure 7.11: Two obstacle views combined for two panels in series.

Both panels in this scenario are losing individual performance due to the series connection. The performance reduction for a panel could be expressed as the difference between its performance with the single obstacle (P_A) and the performance with both obstacles (P_{AB}). The total performance reduction due to this series connection would then be the sum of both individual panel losses. The higher this total loss, the higher the miss match. The relative miss match for two panels can then be defined as:

$$miss_match = 1 - \frac{2 \cdot P_{AB}}{P_A + P_B}. \quad (7.1)$$

The algorithm to determine miss match with method 1 in pseudo code by algorithm 7 in Appendix B.

This method will however only work if both panels have the same or at least almost the same relative intensity. Not any two panels can be compared in this way. If panel A is facing west and has an obstacle in east direction, that obstacle doesn't matter very much. Another panel B can be facing

east, without obstacles nearby. Comparing those in this way would create an east obstacle for panel B, which would matter a great deal. All of a sudden a non significant obstacle becomes very significant. This is however an extreme scenario and for panels on the same roof section this does not occur.

Panels with the same orientation, tilt angle and position will have the same relative intensity. For panels on the same roof, the position difference doesn't really matter for the relative intensity. So for panels on the same roof with the same orientation and tilt angle this method will work. A miss match between multiple modules can be calculated quickly by stacking the obstacle views. For a system with n modules the miss match is defined as

$$miss_match = 1 - \frac{n \cdot (P_1 \cdot P_2 \dots P_N)}{\sum_{i=0}^n P_n} \quad (7.2)$$

This method is tested for a system that consists of 8 panels on the same roof. In figure 7.12 the system can be seen, with three obstacle views of three different panels.

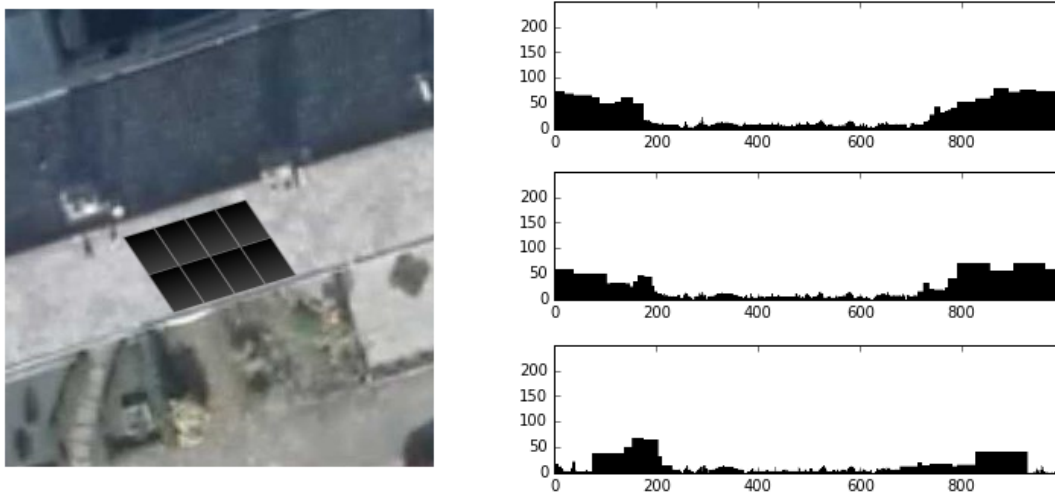


Figure 7.12: A PV system on a pitched roof with the obstacles views from three panels.

The calculated miss match for this system according to the developed method is 1.2%. This is a low number, but there is also no reason to suspect any miss match in this case. In another example there is a dormer next to the panels. This situation can be seen in figure 7.13.

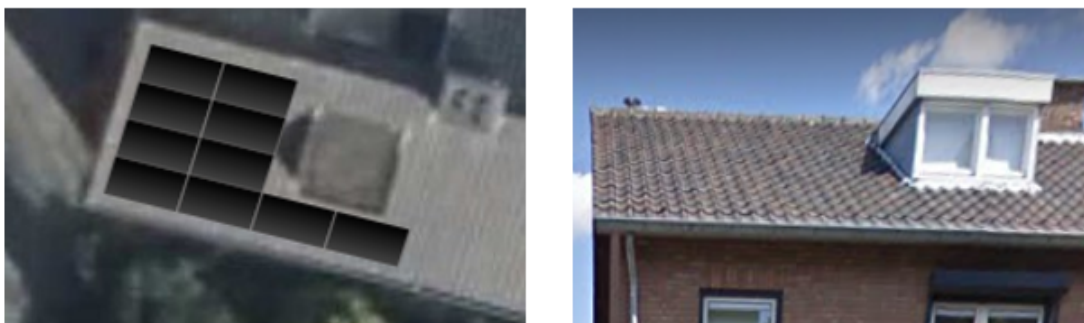


Figure 7.13: A PV system on a pitched roof with a dormer.

The miss match of this configuration is significantly higher than the previous one. The miss match of this configuration is 10.7%. This is a lot higher, but there is a dormer on the roof that causes some shading of the panels.

7.2.3. Miss match method 2

Another method was developed that can be applied more generally and doesn't require the same relative intensities. It is based on the principle that the sun moves around the horizon every day and that irradiance that comes from certain directions will come at certain times during the day.

Obstacles that block the east could be considered as obstacles that block the irradiance in the morning. The obstacle view can be seen as a time line with obstacles that block the irradiance for a certain moment in the day. The sun is however not at the same azimuth at the same time every day. This makes it not possible to take yearly irradiance data per azimuth angle and transform that into irradiance data per time during the day. The sun paths and solar irradiance are all on a monthly basis. A comparison between two panels is done on a monthly basis. In one month the differences between the days are limited and the sun path is almost equal. The relevance of obstacles depends on the month. An obstacle might block the direct sunlight in the winter, but not in the summer when the sun is higher at the sky. This makes it not possible to work with only a single month. For that reason all of the months are used.

The irradiance contribution per orientation is determined by using the relative intensity, the sun tracks and the obstacles. The obstacles block the contributions of irradiance for certain pixels. The diffuse and direct irradiance at each orientation is determined by summing the 250 pixels for that orientation. The total irradiance is then the sum over all the 1000 orientations. This is done for every month and then all the months are summed to get a yearly yield.

The miss match is based on the difference between two different performance calculations: a performance when the performance of the panels is added independently from each other (A) and a performance when the panels are treated as a series connection with a string inverter (B). For performance A the irradiance at each orientation is just summed, for performance B the worst irradiance at each orientation is taken and multiplied by the amount of panels. These different performances are illustrated in figure 7.14

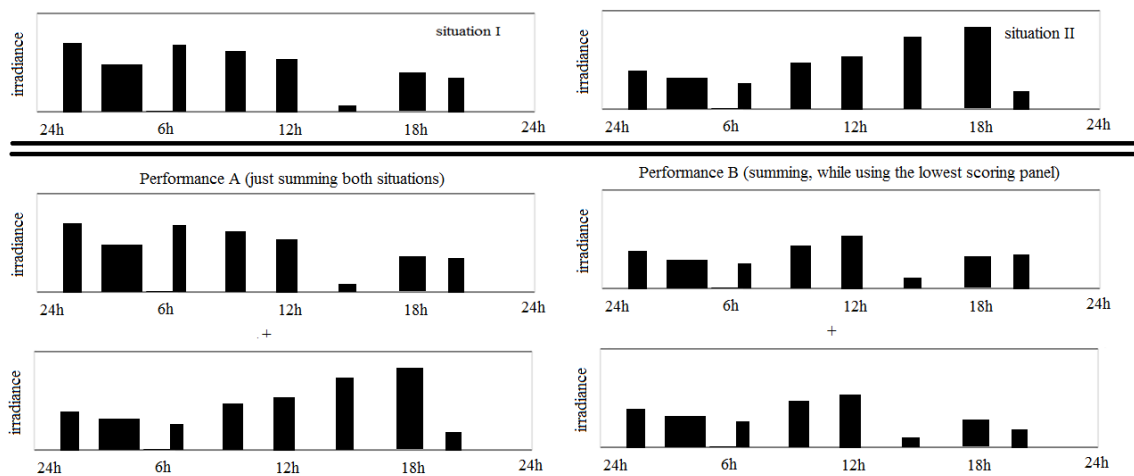


Figure 7.14: Calculation of the two different performances for miss match method 2

The different performances A and B are calculated for each month. Then all the months are summed. The calculation works the same for more than two panels. The miss match is given by

$$miss_match = 1 - \frac{B}{A}. \quad (7.3)$$

The algorithm to determine miss match with method 2 in pseudo code by algorithm 8 in Appendix B.

This method allows the use of different orientations. A hypothetical scenario is created in which two panels are placed at the same location, but with different orientations. One panel has a fixed orientation and the other panel rotates. The miss match is calculated for different orientation differences and the results are plotted in figure 7.15.

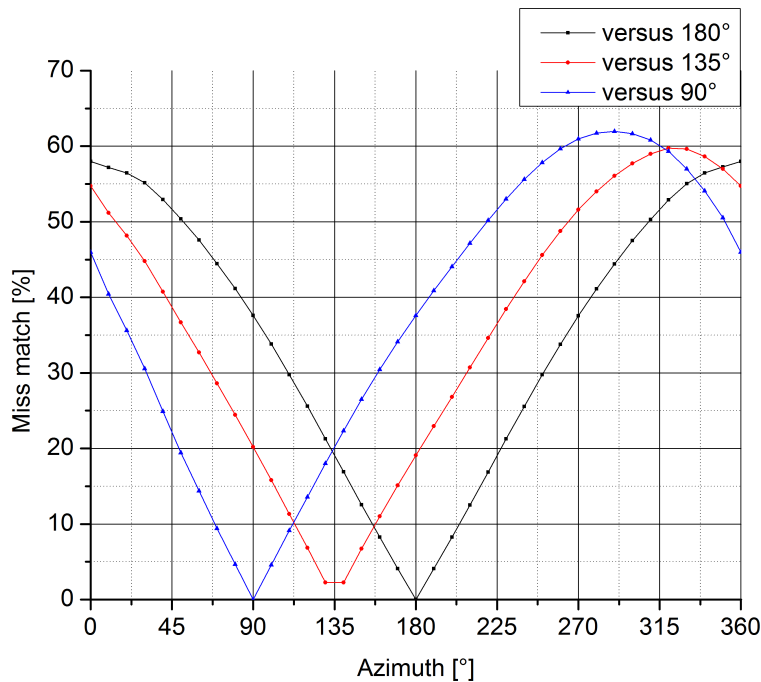


Figure 7.15: Miss match of a panel versus another panel with a different orientation, but on the same position with the same tilt angle of 35 degrees.

The miss match is zero when the panels are facing the same direction, as can be expected. The maximum miss match is however not always with exactly opposing orientations and the maximum miss match value is not always the same. The reason that the maximum miss match is not at exactly opposing orientations is due to the fact that the sun tracks are not symmetrical around the south. The simulation was also done with only the diffuse irradiance and then the maximum miss match was at the exactly opposing orientations. This confirms that the asymmetrical sun tracks are the cause of this shift.

There is also difference in miss match between seemingly similar situations such as 30 vs 330 degrees and 60 vs 120 degrees. This is due to the difference in total irradiance at the different orientations. The orientations 30 and 330 degrees can receive about the same total irradiance, while the total irradiance of the orientations 60 and 120 degrees will differ significantly. This is why they have different miss match values, even though their orientation difference is the same.

A calculation of the same roofs as in figure 7.12 and 7.13 result respectively a miss match of 1.1% and 10.7%. Both methods produce very similar miss match outputs.

The miss match values can determine whether a string inverter or micro inverters are more optimal to use. At a certain threshold of miss match it is better to switch to micro inverters. In order to determine that threshold more research is however required. The next step in this research was to look into how large panel sections can be optimally divided into separate strings, using the developed miss match algorithms. The miss match method 1 is used, as it is faster and works fine for panels on the same roof.

7.3. String building in large PV systems

The total performance of large panel sections divided into separate strings depends on how these strings are configured. Some panels will have a large miss match with each other and ideally those will lie in a separate string. The goal is to find the configuration that will result in the highest performance, so the lowest total miss match. In theory all possible combinations could be made by just connecting the wires behind the panels in a certain way. In practise this might not be the case, as very complex

wiring is something that installers want to avoid and panels close to each other are preferably in the same string. In this section a method to determine the most optimal string configuration is developed.

Method 1: Polygon combinations

The workings of the developed algorithm is explained with the use of an example. A pitched roof section is used and a panel section of 16 panels is placed on it. These panels are all numbered as can be seen in figure 7.16.

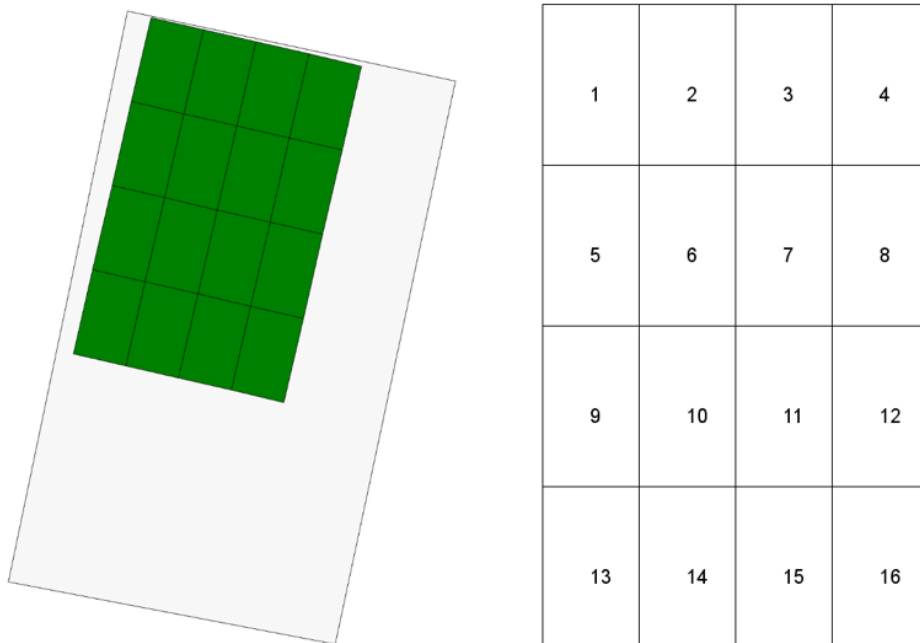


Figure 7.16: A panel section of 16 panels that is used to test the string building algorithm.

The most simple method is just to determine all the possible string configurations and determine which will result in the highest yield. The first step is then to determine all the possible string configurations. There is one constraint for these strings. All panels in the same strings should be physically connected to each other. The connection should be with the sides of the panels above or next to each other, diagonal doesn't count. This is done to have the panels in the same string in an easy to mount configuration and close together.

The algorithm first computes all the possible string configurations for the amount of strings that is desired. For a string of 8 panels out of 16 panels there are almost $1.3 \cdot 10^4$ combinations. This however also includes undesired strings such as (1, 2, 4, 5, 7, 14, 15, 16). All of these bad configuration have to be filtered out first. This is done by first making a multipolygon of the selected panels and then taking the cascaded union. A multipolygon is a collection of polygons. The cascaded union merges all the parts of the multipolygon together. The length of this cascaded union is an indication for whether the panels are all connected. If they panels are all connected together it has a length of one. This is illustrated in figure 7.17.

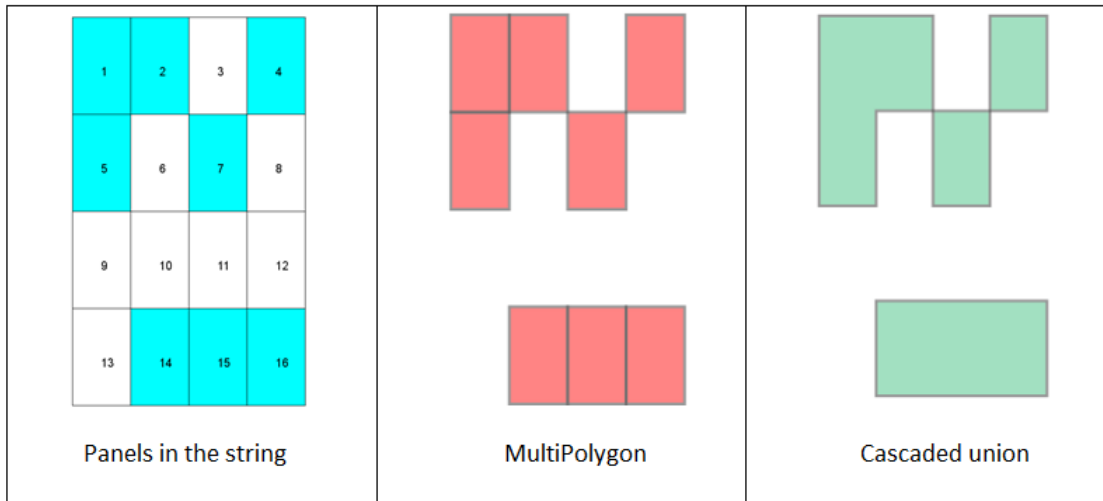


Figure 7.17: A panel selection is transformed to a cascaded union.

Diagonal connections do not count as a connection, so in the example of figure 7.17 the length of the cascaded union is four. This method should however be adjusted if panels in different rows on flat roofs are allowed to be connected with each other, because they don't physically touch each other. With this method all the invalid string options are filtered out, leaving only about $1.5 \cdot 10^3$ valid strings. Now all the possible combinations of two strings are considered, which is an enormous amount of about $1.1 \cdot 10^6$ possibilities. There is however a restriction that only the combinations that have all the panels included exactly once are valid. This leaves only 70 possible string configurations. The miss match of all these configurations is calculated and the best configuration is chosen.

This same method was also done for string configuration with four strings. To get an idea of the computation speed, these calculations both took about 16 seconds. The string configurations with the lowest miss match are indicated in figure 7.18.

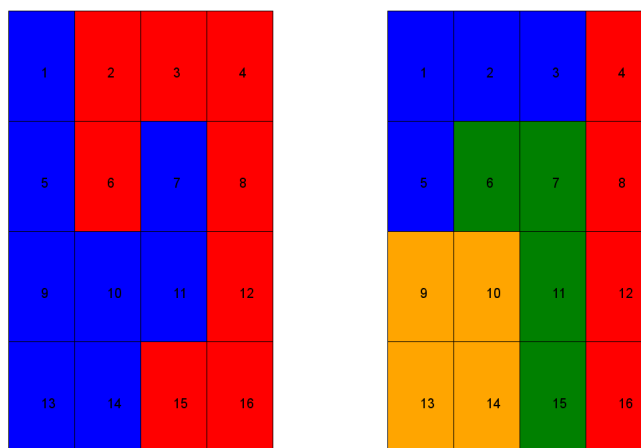


Figure 7.18: Optimal string configurations for two and three separate strings for a system with 16 panels.

These configurations were checked by performing a performance calculation for all the string configurations. This was done by doing a calculation for the separate strings and then adding the values of the strings together.

For the two string configuration the prediction was correct. There were however slight differences in the ranking of the different configurations. In the configuration with four strings the second best predicted configuration turned out to be the best. The difference is only less than 0.5%. The top 10 configurations was the same, but with some order differences. It seems that the prediction is accurate in determining the order with a slight uncertainty of less than a percent. More test will have to indicate

how the string determination performs on a larger sample of scenarios.

This method is however not practical for very large systems, as the amount of possibilities to check becomes too large. A solution for this is to split a large section into several smaller sections first. This idea was tested for a system of 126 panels. With the standard approach this amount of panels divided into strings of 6 panels gives over $4.9 \cdot 10^9$ combinations of six panels in a string. Checking all these combinations for valid string is not practical. Instead of doing that the panels are first divided into 7 parts of 18 panels. Each part of 18 panels then has about $1.9 \cdot 10^4$ combinations to make a string of 6 panels. This is a number that is more manageable, even 7 times. These 7 sections are then divided into three strings with the standard method. The panels, the way the sections are divided and the final string configurations can be seen in figure 7.19.

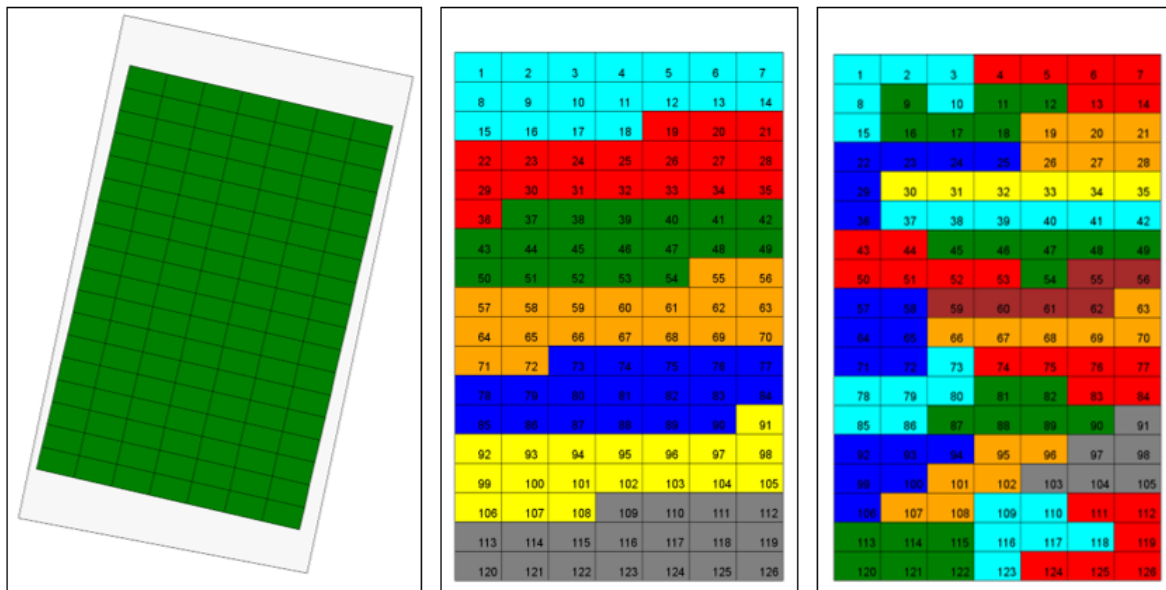


Figure 7.19: A panel section of 126 panels divided into 7 parts and then into 21 strings.

With such a method a large panel section will scale almost linearly, which is a lot better than to keep using the standard method for the entire panel section. There will however be the question of how to divide the sections.

It can be the case that PV system have the same large panel sections. The possible string configurations can then also be predetermined. If a certain panel section then occurs, a predetermined string configuration can immediately be used. If there aren't too many of them, all the miss matches can be calculated up with the best configuration. Another approach would be to completely change the way that the valid string options are determined. A possibility is to look at graph theory.

Method 2: Graph theory

A lot of situations can be represented by a set of points and lines connecting certain points together. Such a mathematical representation of a situation is called a graph. In such a graph the points are called vertices and the lines are called edges [29]. Each panel could be represented as a vertex and a connection between two panels can be an edge. A walk is a way of getting from one vertex to another, a path is a walk in which no vertex appears more than once [30]. A string of panels can be represented as path. An example of how that could look like for the string configuration of figure 7.18 can be seen in figure 7.20.

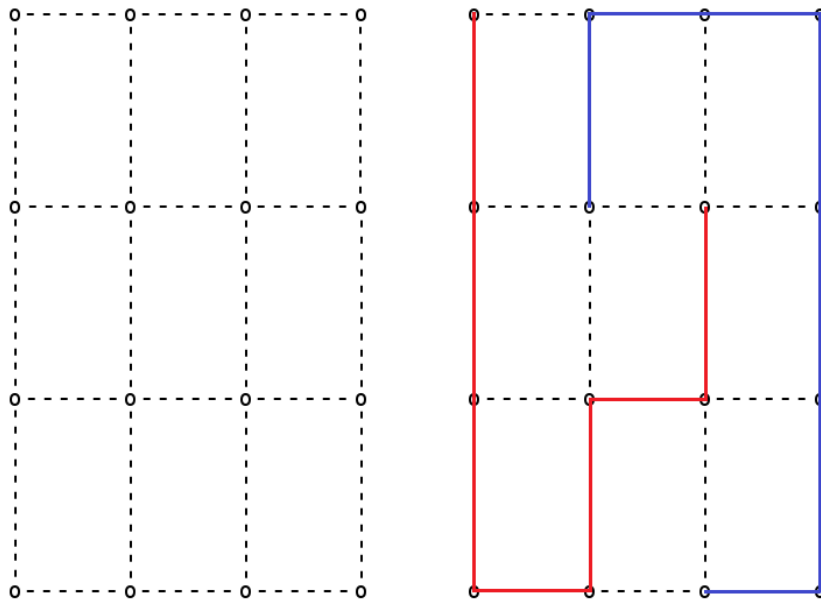


Figure 7.20: A panel section represented as a graph, with strings as paths.

Certain algorithms could be looked up in literature that find all paths of a certain length, with the restriction that all connections have to be either horizontal or vertical. Such an algorithm might make it possible to find all the valid strings. It can however still result in too many possibilities to all calculate.

It is also possible to determine the miss match between adjacent panels. This could be used as a weight between two panels. All of the miss matches between adjacent panels are calculated for the same 16 panel example and a new graph is created, but with the miss matched indicated at the edges. This graph can be seen in figure 7.21, together with the two solutions for two and four string configurations.

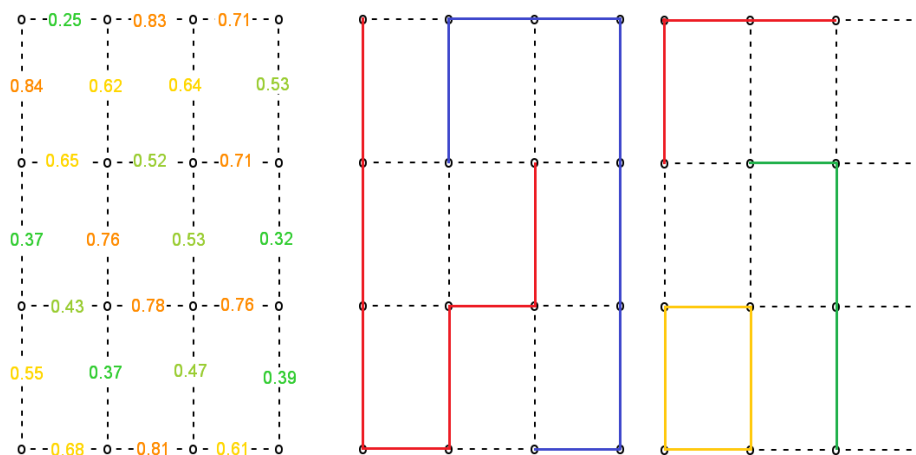


Figure 7.21: Weighted graph with strings as paths and miss matches as weights between connections.

Assigning a weight to a certain edge is common practise and will result in a weighted graph [30]. There are techniques that can find paths of certain length with the lowest cost based on these weights. This is however beyond the scope of this project and is just an idea that can be explored in the future. If a cost function can be made based on the miss matches between the panels, an algorithm could be developed that would find a collection of paths with the lowest total cost. That would then be the optimal string configuration.

7.4. Optimal string inverter sizing

Most of the time a PV system will not operate at its peak power. The performance of the PV system with respect to the rated power is dependent on several aspects, such as the irradiance, the temperature and the orientation and tilt of the solar panels. In 2015 a master student named Jessica Hernandez has done her master thesis on the optimal sizing of the inverter based on the installation characteristics of PV systems. Graphs were made that indicate at which power levels a PV system operates. The power is shown as a blue bar chart in which the height of the bar represents how long the system has operated at that power level. A red line indicates the total energy that the system has produced while operating at that particular power level. Such a chart for a system facing south can be seen in figure 7.22.

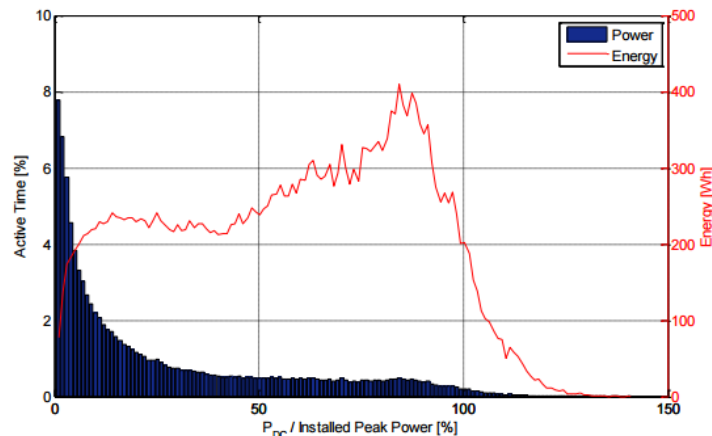


Figure 7.22: DC power and energy distribution of a PV system facing south. Image taken from [31]

It can be seen that for a PV system facing in the most optimal direction is almost never operating at its peak power. A system facing north will have a different graph. This graph will have an average operating power that is much less than a system facing south. A graph for a system facing north can be seen in figure 7.23.

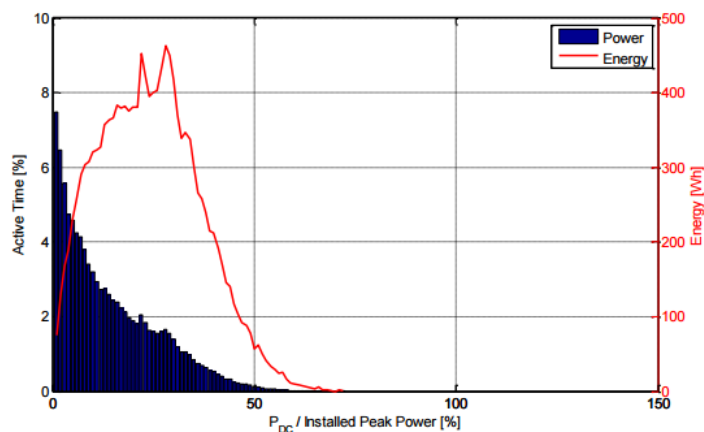


Figure 7.23: DC power and energy distribution of a PV system facing north. Image taken from [31]

For a PV system facing north, the performance is significantly worse. The PV system is never operating at its peak power, which means that an inverter with a nominal power equal to the peak power of the system is not necessary. The research resulted in a table which gives the optimal inverter sizing for a PV system with panels with a certain azimuth and tilt angle. This table can be found in Appendix C. The inverter sizing ratio that is displayed is defined as

$$R_S = \frac{P_{peak}}{P_{nominal}} \cdot 100\%, \quad (7.4)$$

With P_{peak} the peak power of the panels and the $P_{nominal}$ the nominal power of the inverter. The table is based on the fact that PV systems rarely operate at their peak power.

The thesis concludes that often times an inverter can be chosen that is smaller than the peak power of the PV system. This will result in a reduction in cost, as well as an inverter that operates at a higher efficiency. This last aspect is due to the fact that the inverter efficiency drops significantly if its input power falls below about 20% of its nominal power. If a smaller inverter is chosen, the average operational power will shift towards a larger fraction of the peak power. This will increase the efficiency of the inverter. This concept was investigated by Hernandez and an optimization function was written to find the optimum inverter size based on the installment characteristics of the PV system.

The optimization function developed has a peak for a certain inverter size. It is however not always possible to pick that exact inverter size. Sometimes a slightly larger or smaller inverter has to be chosen. The table shows an inverter range around the optimal inverter size. Picking an inverter size inside that range will result in a performance that is within 1% of the optimal inverter size.

The table has only a finite amount of azimuths and tilt angles. Often times the actual azimuth and tilt angle of the panels in the designed systems will fall in between certain values. An interpolation can then determine the right correction factor.

All of the inverter reduction ranges are examined for two cases. The first case is for all the azimuths and a tilt angle of 30 degrees and the second case is for all the tilt angles for an azimuth of 180 degrees. These ranges can be seen in figure 7.24

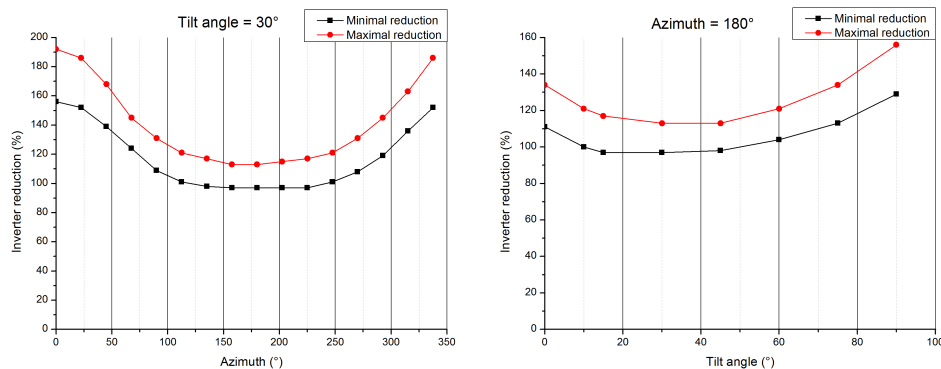


Figure 7.24: The inverter sizing ranges for different installation characteristics.

The graphs in figure 7.24 indicate that a linear interpolation between the ranges can be a reasonable approach to get an inverter range for all possible azimuths and tilt angles. The ranges were first interpolated to correct for the azimuth if necessary and then an interpolation was done for the tilt angle if needed. The result is that an inverter size range is given based on the azimuth, tilt angle and total Wp of the panels. This could later be used to automatically pick an inverter to match the automatically placed panels. In the final algorithm the code used by Hernandez can be implemented in the algorithm, instead of interpolating from a table.

8

Proof of Concept

During the early stages of the development, the algorithms were tested on artificial roof scenario's. In the final stages of the development the algorithms were however tested on real roofs. The algorithms were also tested on real roof scenario's, to see if they have potential. This chapter explains how the algorithms were tested, how they were evaluated, how the data was obtained and how the results were produced.

8.1. Methods of comparison

The first thing to do when testing the algorithm is to establish what the algorithm is trying to accomplish. The second thing is know with what the results of the algorithm can be compared.

The goal of the algorithm is to design PV systems automatically. The algorithm also tries to make the designing process faster than is manually possible. Lastly it tries to design PV systems that are better than the ones that are manually designed. All of these objectives mean that the algorithm has to be compared with a manual designing process.

Unfortunately there isn't really comparison material at hand. A good and objective comparison would require the algorithm to be compared with designs on the exact same roofs with the exact same design preferences and actually also without the manual designers knowing that their design will be used as comparison material. Such an objective comparison setup was not practical and also not required for the prototype testing of the algorithm. Only a proof of concept is done to see if the algorithms have practical potential. A simpler comparison is done instead.

Many PV systems have been designed with the use of the Solar Monkey software. This means that they have a database with all kinds of designs that were made by installers using the software. These designs are considered manual designs, even though they are made using software. The algorithm was set to find solutions for the same roofs as these designs. Two things should however be mentioned about these designs:

- It is unknown how much effort and time was put in the design

The only thing that is known is the design, but it isn't known how much time was put into designing it. It is also not known whether the design had actually been build on a roof or whether it was just a tryout.

- It is unknown what the intention of the designer was

There are scenario's where a certain design is made, but more panels could probably fit on the roof. It is then not sure if the designer did not notice that, couldn't make a design that would fit more panels or simply didn't want more panels.

The second point is a big concern for the settings of the algorithm. To stay away from too many assumptions, the algorithm was set to just find a solution for the same amount of panels that are in the manually designed system. The size of the panels was kept the same as was used in the manual design. The maximum panel placement algorithm as well as the finite panel placement algorithm were tested on both flat and pitched roofs.

To gain more insight about the range of possible solutions that the finite panel placement algorithm can present, some roofs got a different design preference. Not the exact amount of panels as in the manual design had to be placed, but one panel fewer or more was also alright. This is a design preference that is very likely to occur when the final algorithm is used.

For each algorithm that was tested, a few evaluation aspects were set up. These evaluation aspects are given in this overview:

1. Finite panel placement algorithm for pitched roofs and flat roofs

- Finding the manual design and/or reasonable alternatives

The algorithm can be said to perform well if it finds at least a few reasonable solutions that fit the design preference. With reasonable is meant that the panel layout is deemed practical. Often times this will include the design that was manually made. How often these solutions are found is measure of how successful the algorithm is.

- Differences in the performance between the designs

Ideally the algorithm can find designs that perform better than the manual designs. Even if there is no performance difference or the algorithm produces slightly worse performing designs, it can still be viable algorithm due to the fact that the designs are made automatically.

- Algorithm duration compared to manual designing

The duration of the algorithm will be a factor in determining in what kind of application it can be used. The goal is to eventually have it much faster than the manual designing process.

2. Maximal panel placement algorithm for pitched roofs and flat roofs

- Azimuth differences algorithm and manual designs

The algorithm for flat roofs determines the orientation of the panels by using the roof polygon. Comparing this with the azimuth that the installers had chosen to align the panels with the roof will indicate how well this method works.

- Differences in the amount of panels

Automatically placing the maximum amount of panels could use all the available roof area more optimal and result in more panels than were placed manually. The difference in the amount of panels placed is a measure for how well the algorithm performs in this aspect.

- Algorithm duration compared to manual designing

Placing a maximum amount of panels is often done for flat large roofs. This can take quite some time to design and an automatic designing process can really make a difference here, especially since that algorithm doesn't seem to require much computational power. The goal is to eventually have an algorithm that is significantly faster than the manual designing process.

3. Finite panel placement algorithm for a panel range

- Diversity among the produced solutions

The algorithm should provide different solutions that all fit the design preference. They should be diverse, which means that they have different layouts or number of panels. This gives the user of the algorithm several choices for their design.

- Algorithm duration compared to manual designing

This design preference will likely be what is used in the final algorithm. It will take longer than looking for solutions with just a single amount of panels. It is important to get an idea of how the computation time scales with this other design preference.

All of the roofs had to be collected manually, as will be discussed in the next sections of this chapter. This made gathering large amounts of roofs impractical. It was also not deemed necessary to collect a lot of roofs, because the testing only serves as a proof of concept and not as a statistical analysis of the performance. A total of 71 roofs were used to test the algorithms.

8.2. Extract roof sections and obstacles

In order for the algorithms to run for a certain roof section it requires a roof surface polygon, as well as polygons for any obstacles on the roof. These are not just available and obtaining these polygons has to be done by hand. This is done with free software called Quantum Geographic Information System (QGIS). This is software that enables you to view and edit geospatial data. Several different layers can be used to display data.

Necessary for the extraction of roof surfaces is the AHN data. This can be downloaded from the Dutch website of the Publieke Dienstverlening Op de Kaart (PDOK). The AHN data is very large and the Netherlands is divided into small blocks that can be downloaded separately. The latitude and longitude coordinates of the locations of a sample of the manually designed PV systems from the Solar Monkey database were loaded into QGIS, which creates a layer with points that have coordinates. The available AHN blocks that can be downloaded can be seen on a map. Both the locations of the PV systems and the AHN blocks can be seen in figure 8.1.

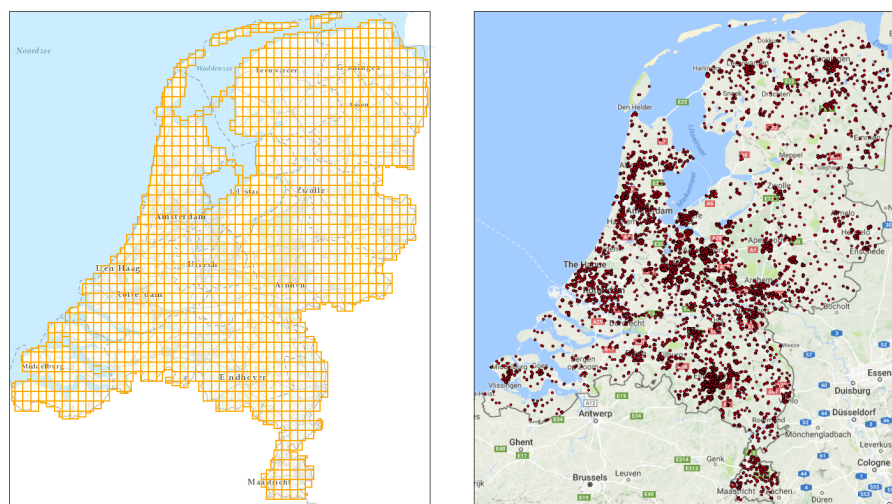


Figure 8.1: Blocks of AHN data that can be downloaded and a sample of PV system locations.

The process of the getting roof surfaces is to download an AHN block which contains a PV system and then to load that AHN data into QGIS. The AHN data then gets loaded into QGIS as a separate layer. The AHN layer and the PV system location points layer can be stacked and can be seen in figure 8.2.



Figure 8.2: The AHN layer in QGIS with the red dots representing roofs with PV systems designed with the Solar Monkey software.

The next step is to select a red dot and look up the corresponding PV system. If that PV system is deemed good enough to use for the evaluation, the roof polygons are created. Some PV systems are obviously only software tryouts and are not fit to use for the evaluation. The creation of the roof polygons is done by creating a new layer in QGIS and to draw a polygon on that, following the shape of the roof on the AHN layer. By following the AHN layer it is made sure that the sizes of the polygons are drawn at the correct scale and at the correct position. It is however difficult to distinguish the roof when looking at the AHN layer. Figure 8.3 shows a certain PV system and the corresponding location on the AHN layer.



Figure 8.3: A PV system designed with the Solar Monkey software and the corresponding AHN layer.

It is possible to recognize the roof sections, but it is difficult to see where the exact peak of the roof is. It is also difficult to see where the roof sections end and to recognize the dormer. The dormer is in this case somewhat visible, but windows for example would not be visible at all. The fact that the roof is pitched should also be taken into account when estimating the polygons of an obstacle. Another difficulty is that the aerial photographs are often taken at an angle, this makes it more difficult to find the exact shape and position of roof obstacles. An attempt is still made to draw the roof polygon as accurate as possible. This is done with the Quantum Aided Design (QAD) plugin from QGIS. Figure 8.4 shows the process of drawing the polygons.

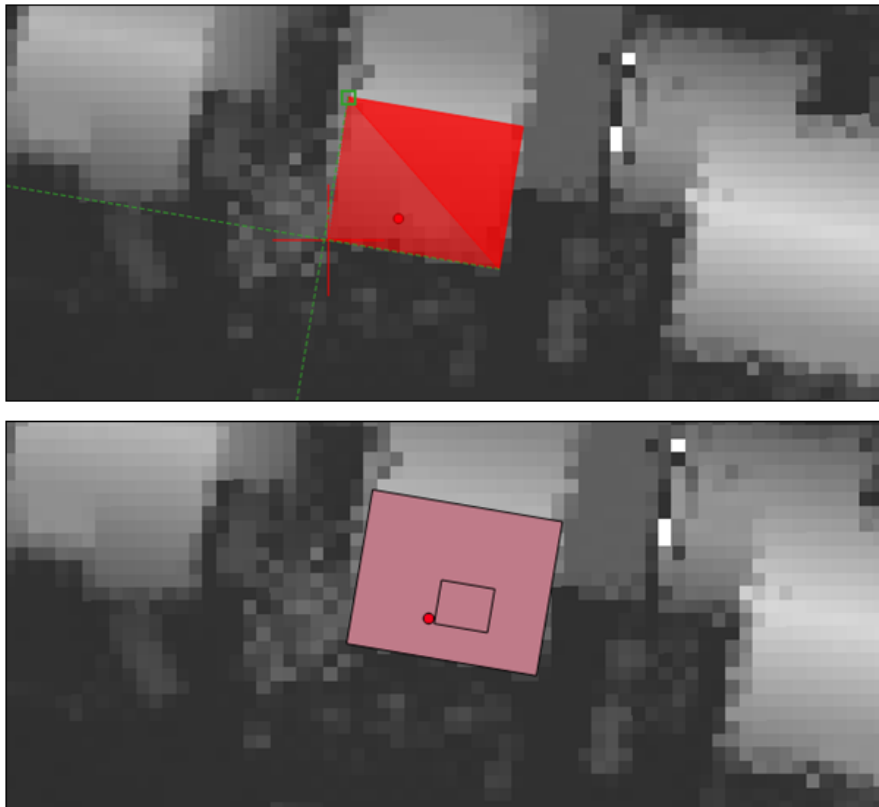


Figure 8.4: Drawing the roof polygon and dormer in QGIS on the AHN layer.

Once the polygons are drawn they are exported from QGIS in the form of a shapefile. The obtained files consists of polygons. Since the polygons are drawn by looking at the AHN layers, the shape will deviate a bit from the actual size. The location of windows and dormers can deviate, because they are very difficult to see on the AHN layer. This introduces a certain inaccuracy. This inaccuracy is also addressed in chapter 9. This method for obtaining the roof polygons is also done for large flat roofs and it then becomes a time consuming process with a lot of estimations of where the obstacles are. To illustrate this an example of such a roof is shown in figure 8.5.

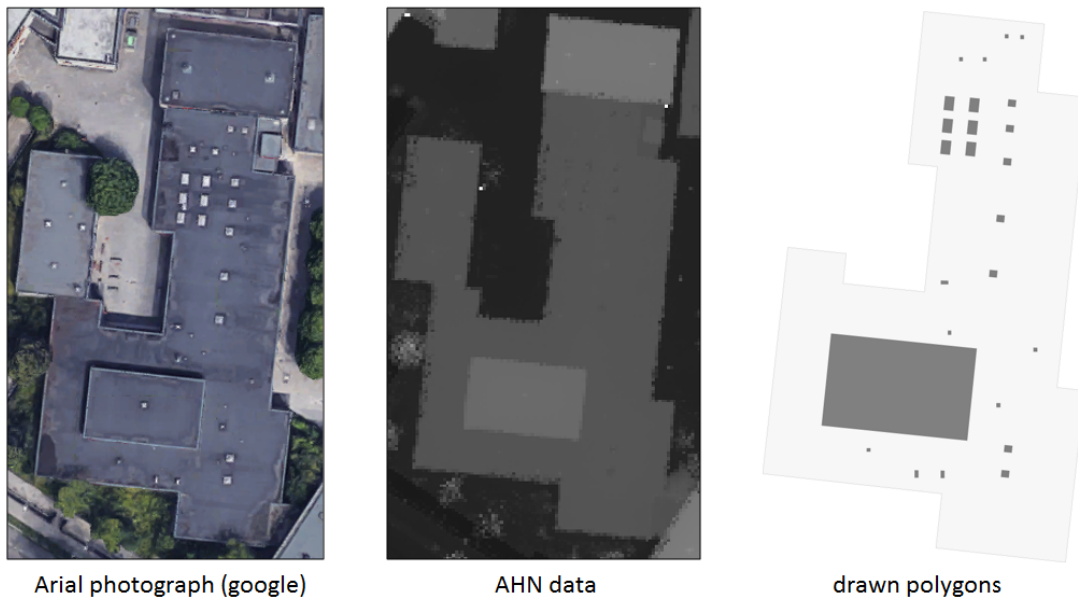


Figure 8.5: A very large flat roof, with corresponding AHN layer and drawn polygons.

Some of the roof obstacles can be seen on the AHN layer, but only barely and most of them are not visible at all. This makes it hard to accurately obtain the roof polygons for a good comparison.

8.3. Performance grid and shape corrections

The next step is to determine a performance grid. To do that the positions of the grid squares have to be determined. More specifically, the center point coordinates have to be determined. This is obtained by placing a grid on top of the roof surface, with the square sizes equal to the AHN data resolution.

The shapefile has coordinates in the World Geodetic System 84 (WGS84), many other coordinate systems can be chosen however. These WGS84 coordinates are in degrees latitude and longitude. To get a proper sizing of the grid, these coordinates are converted to a different coordinate system. The coordinates are converted to the Rijksdriehoeks coordinates (RD coordinates), which is a Dutch coordinate system. This coordinate system has a bottom left origin as can be seen on the right side of figure 8.6.

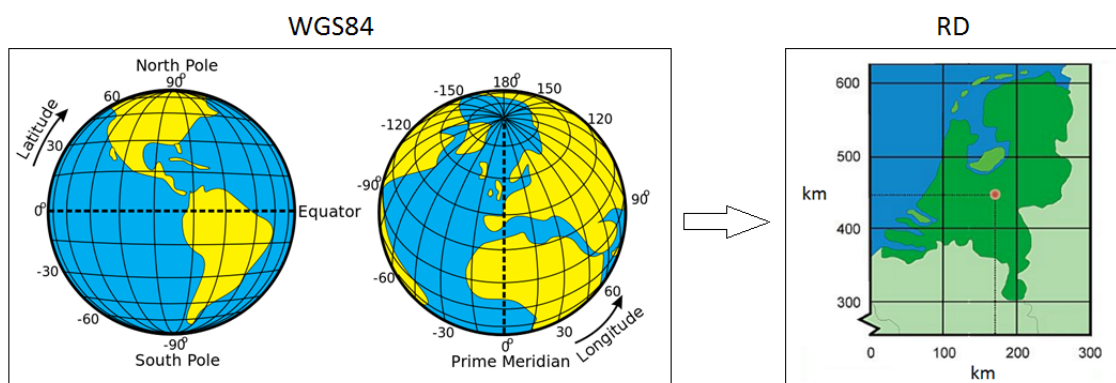


Figure 8.6: The WGS coordinate system and the RD coordinate system. Images adapted from [32] and [33] respectively

The origin used to be in Amersfoort, the location of the red dot. This was changed to have the entire Netherlands in positive coordinates and to have the y-coordinate always longer than the x-coordinate. This is convenient coordinate system, as distances can be easily calculated. The WGS84 coordinates

are converted into RD coordinates with the use of the pyproj library in python. This will result in coordinates in meters to the origin.

A grid with a square size of 50 cm by 50 cm is then drawn on top of the roof surface. The center points of the squares that don't fall inside the roof polygon or that overlap with an obstacle are discarded. The grid is in RD coordinates and the center point coordinates are converted back to WGS84 coordinates. These coordinates are then written to a file and that can be loaded in QGIS to check whether the coordinates are correct. The grid drawn in python and the resulting coordinates in QGIS can be seen in figure 8.7.

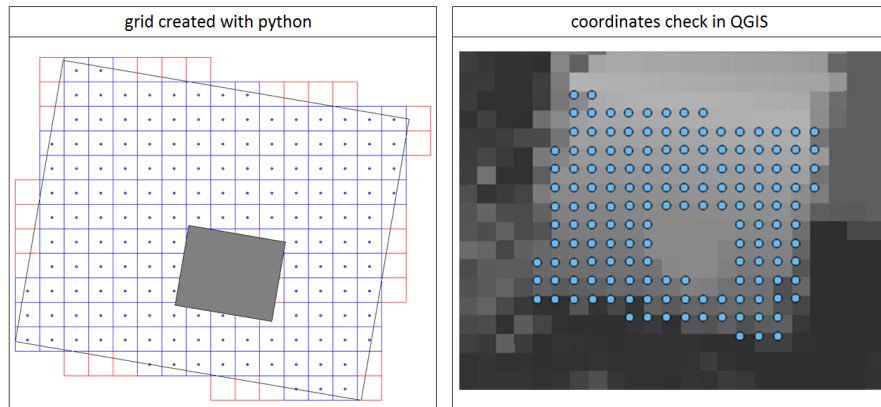


Figure 8.7: A grid drawn with python and the coordinates shown on the AHN layer in QGIS.

The performance is calculated for each point shown in figure 8.7. This is done with the Solar Monkey software. It calculates the performance of a single panel placed at such a point for the pitch angle and orientation of the roof. These values are then used for the performance grid.

Before the calculated values are placed into the grid and the algorithm can run, a few shape corrections have to take place. The roof is drawn based on the AHN layer, which is a top view. This means that the roof section that is drawn is not the actual size of the roof surface. This concept is illustrated by figure 8.8.

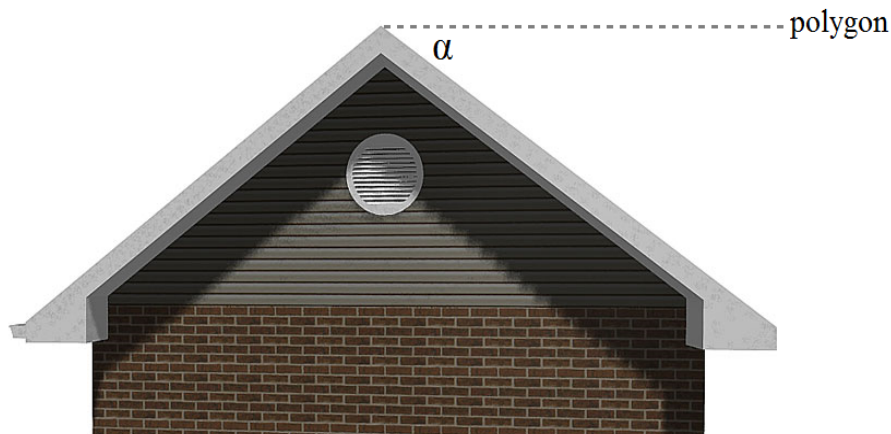


Figure 8.8: Difference between the drawn polygon and the actual roof.

As can be seen in the figure above, the polygon will not be the actual surface of the roof. In order to correct for this the polygon is scaled in the direction of the orientation of the roof. The roof is scaled by a factor A given by

$$A = \frac{1}{\cos(\alpha)}. \quad (8.1)$$

Another correction has to take place in order for the panels to be placed in the correct orientation. The panels have to be placed with either the long or the short side horizontally. The polygons are however drawn in the orientation that the roof was facing. The polygon is rotated in such a way that the front of the roof is facing the viewer. The roof is rotated first to correct for the azimuth and then stretched to correct for the pitch angle. The rotation is based on the azimuth that the panels have in the manually designed system. An overview of the operations on the imported shape file polygon can be seen in figure 8.9.

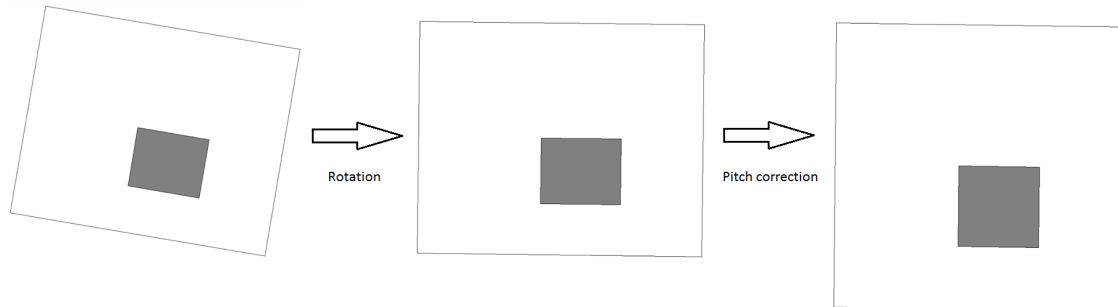


Figure 8.9: Roof azimuth and pitch corrections.

These operations are also done on the performance grid, so that this will fit on the newly formed roof shape. The performance values that were calculated are placed in the performance grid. Only the points that fell within the roof polygon were used, which means that some squares do not have a value. To obtain a value for these squares an interpolation is performed. A value for the missing squares is determined by taking the average of the surrounding squares. The obtained grid and the grid after interpolation can be seen in figure 8.10.

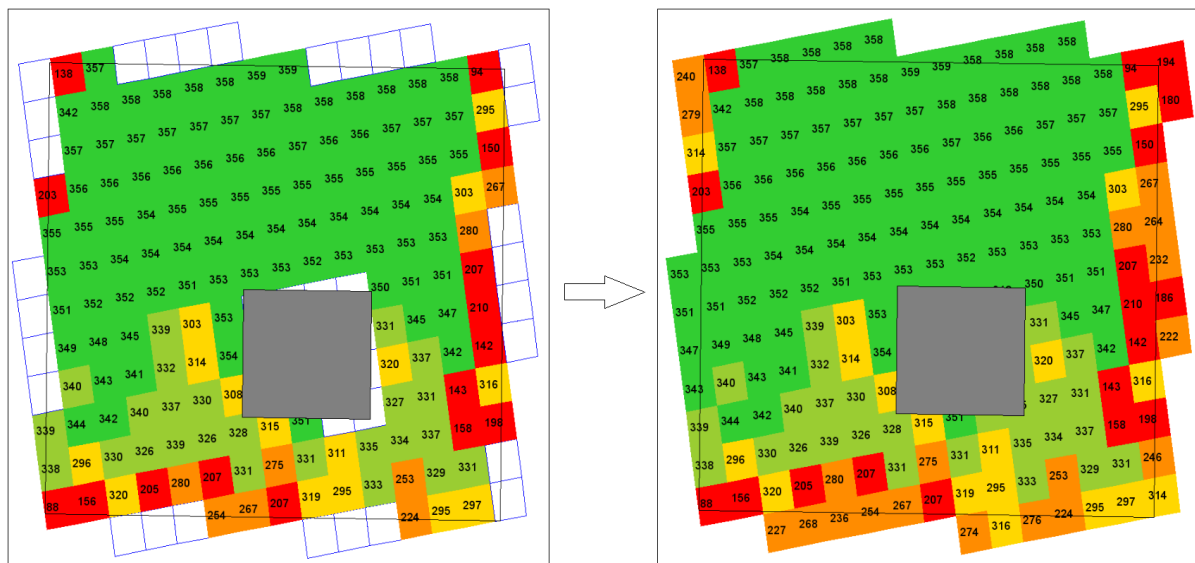


Figure 8.10: The calculated performance grid with interpolation of the missing values. The values represent the performance in kWh/year that a panel of 275 Wp will have at that position.

The grid is still rotated with respect to the roof, because it was drawn directly on the roof surface prior to the rotation. It does however not matter for its purpose that it is rotated. The performance values in the grid represent the yearly yield in kWh of a single PV panel at that position.

It should be noted that calculating the performance of string in series with the performance grid will result in a slightly different performance than when the system is calculated by the Solar Monkey software. In that software the obstacles views of all the panels will be stacked, in the algorithm the worst performing panel will be used multiple times. This is not the same, although the difference will often be very small.

The values at the edges are significantly smaller than those in the center. The reason for this is that the calculation is based on the AHN data and the height at the edges can be a lot smaller, it can be just off the ledge of the roof. As was mentioned earlier, it is difficult to precisely draw the roof polygons and some edges can therefore be not entirely on the actual roof. Other values that are lower are due to the shadows created by surrounding obstacles, such as the dormer or buildings and trees nearby. The corrected roof polygons and performance grid form an input for the algorithms. The algorithm settings used in the validation are discussed in the next section.

The algorithm obtaining the polygons and performance values to be used in the algorithms is given in pseudo code by algorithm 6 in Appendix B

8.4. Algorithm settings

The current prototype of the algorithms have several settings that can be adjusted. A single setting was chosen for all the roofs. These settings can be optimized after studying the results. The settings for the finite panel placement algorithm is discussed first.

8.4.1. Finite panel placement algorithm

Each of the four stages in the algorithm have settings that can be adjusted. All of these settings have been picked based on experience during the development of the algorithm. The final settings will have to be determined after testing on a large scale. The flat roof algorithms also had to determine the tilt and the azimuth of the panels, but those were simply kept the same as in the manual design.

Roof positions

The amount of roof positions depends on the distance between the roof points, also called the step size, and the distance from the edge of the roof that is left clear. The distance that is kept clear from the edges is called the buffer. There are three steps in the algorithm and all of them require different roof positions. The starting points for the trivial and combined solutions are created by using bounding box of the roof polygon. A grid of points is created with distances between them equal to the step size and only the points within the roof polygon and outside the buffer zone are kept. This is an overview of the parameters chosen for each step:

- Trivial solutions
 - The buffer is half the size of the smallest edge of the trivial panel sections
 - The step size is 25 cm.
- Large solutions
 - The centroid of the roof polygon is taken and from there 15 steps of 10 cm in all directions are taken. The centroid is geometric center of the polygon shape.
- Combined solutions
 - The buffer is the size of half the width of a single panel
 - The step size is 25 cm

Determine panel sections

The panel sections are based on the roof dimensions, the design preference and whether there are obstacles on the roof. The first step is to determine out of which panel sections the solutions can be build, this can be a single section or a combination of two sections. The panel sections that can never fit on the roof polygon are discarded. This is determined by comparing the length and width of the panel section by the bounding box of the roof polygon. Only rows with more than two panels are considered. If the roof has obstacles, the list of panel sections is expanded by adding larger panel

section. Each section is taken and two extra panel sections are created out of it. A panel section $[a, b]$ produces the sections $[a + 1, b]$ and $[a, b + 1]$. All the duplicate sections are removed.

Each step of the algorithm uses a different set of panel sections. The panel sections used in each step can be seen in this overview:

- Trivial solutions
 - The trivial panel sections that fit on the roof
- Large solutions
 - The largest panel sections that fit inside the bounding box of the roof polygon, minus an edge of 10 cm
- Combined solutions
 - The panel sections that could make a solution consisting of two sections

For a flat roof there are some changes to the panel section selection. For an east-west configuration only panel sections consisting of a 2 by X configuration were chosen. This configuration is a row with two panels back to back and the row is X panels long.

Place sections and build solutions

Only the sections from the combine solutions step are combined with each other. The settings in this stage are pretty much fixed for this phase of the development. The only settings that can be changed are the amount of panel sections that are combined in a single solution and whether the overlap between panel sections is allowed. During the testing only solutions with two sections were made and no overlap between the panel sections was allowed.

Evaluate and rank solutions

The evaluation and ranking of the solutions was done based on the multi objective optimization described in chapter 6.

8.4.2. Maximum panel placement algorithms

The maximum panel placement algorithm has the step size as the most relevant parameter. This step size was set to 10 cm during the validation. For a flat roof the row distance is also an important parameter. This one was set to be the same as the row distance used in the manual design.

9

Proof of concept results

In this chapter the results of the algorithm testing is discussed. Too many roofs have been evaluated to show them all individually. Some will be used as examples to point out certain interesting findings, all of them are used to determine general trends. Each algorithm with its evaluation aspects is discussed in the next sections and at the end of this chapter a summary of the results is given.

9.1. Finite panel placement algorithm for pitched and flat roofs

After looking at the first results it was realized that the algorithm did not find solutions for scenario's that seemed straightforward. The cause of this is the inaccuracies in roof shapes that arise when drawing the roof polygons. This makes it so the required amount of panels sometimes doesn't fit on the roof, because the roof shape is drawn a bit smaller than it in reality is. Sometimes the roof is drawn too big, although this is less of an issue for finding valid solutions. Both situations make the comparison invalid, because the amount of panels that actually fits on the roof is different when the algorithm is used.

It was decided to adjust the panel sizes slightly to make sure that at least the required amount of panels would fit on the roof. This was done to make sure that the inaccurate roof shapes would not interfere with the evaluation of the algorithm. The roofs were tested with and without the panel adjustments. That the panels were slightly adjusted doesn't matter for the conclusions about the workings of the algorithm, it does tell something about the relevance of the drawing accuracy. A total of the 54 roof sections were tested for this algorithm and 57% of them had the panels adjusted. The width and length of the panels had to be adjusted about an equal amount of times. The average length correction was 18 cm and the average width correction was 9.3 cm, with average panel dimension of 166 by 100 cm.

9.1.1. Finding the manual design and/or alternatives

The algorithm has three steps in the search for solutions. The final combined solutions step was not applied to all of the roof sections. This was done, because the combination of roof size and the amount of panels would result in a very long computation time. In such scenario's a solution would likely be found by just using the trivial and large solutions steps. Such a judgment call should be made by the algorithm in the future, but that part was not yet developed at that time. The pitched roofs are discussed first.

Pitched roofs

Of the 54 roof sections a total of 41 roofs were tested with steps of the algorithm, with and without adjusting the panel size. The amount of times that the exact solution of the manual designer was

found improved when the panels were adjusted. This is visualized in the charts in figure 9.1.

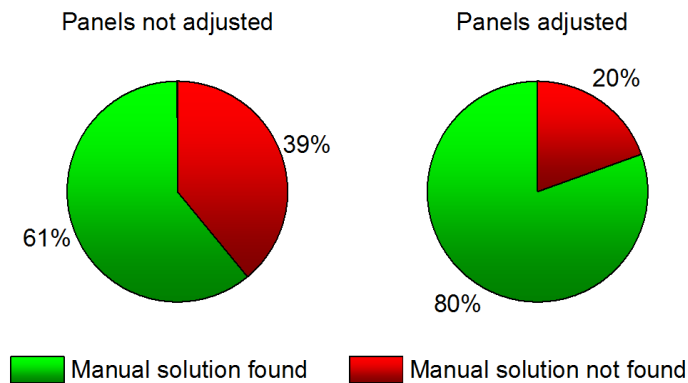


Figure 9.1: Percentages of how often the algorithm found the manual design, with and without panel size adjustments.

When the exact solution wasn't found that doesn't mean that there was not a useful solution. When the exact solution was not found, a good alternative was always found when the panel sizes were adjusted. Without the panel size adjustments it was sometimes the case that no solution were found, this happened with 17% of the roofs. This is because the required amount of panels simply did not fit on the roof for the section layouts and positions that were considered. The cases in which solutions were not found almost always had obstacles on the roof. A reason for this is the inaccuracy in drawing the obstacles. They were often shifted a bit, which made certain layouts no longer possible.

The layout for a roof without obstacles is almost always just the trivial solution, except when the roof doesn't have a simple rectangular shape. An example of such a roof and some of the solutions can be found in figure 9.2.

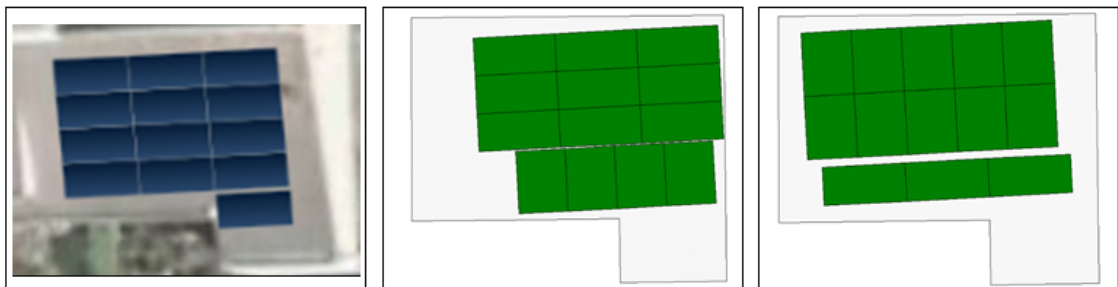


Figure 9.2: An example of a roof without obstacles that requires a non-standard solution

In a scenario such as in figure 9.2 a trivial solution can sometimes not be found due to the shape of the roof. The large sections that are searched for are based on the bounding box of the entire roof polygon and it can happen that the sections based on that don't fit well enough. This was the one scenario where the algorithm didn't find the designers layout.

Another reason that certain solutions are not found is due to the step size of the algorithm. It can happen that a row of panels will fit on the roof, but the algorithm doesn't find it due an unfortunate step size. This concept is illustrated in figure 9.3.

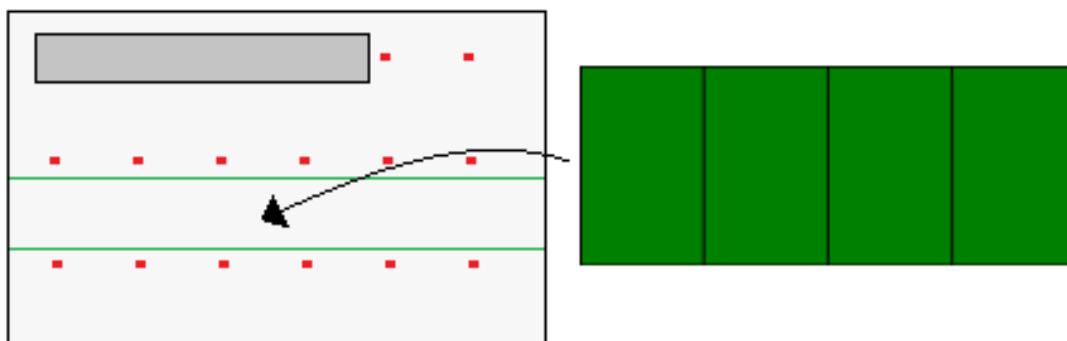


Figure 9.3: A panel section doesn't find a place, because the starting positions are not quite in the right spot.

In figure 9.3 the red dots are the roof positions of the panel sections. A one by four panel section in portrait orientation would fit on this roof if the center points are in between the green lines. The grid of roof positions is however in such a way that the roof points are just outside the zone where the panel section would fit. This particular situation will result in the algorithm not finding this option. This could be prevented by decreasing the step size of the roof positions, but that will have consequences for the duration of the algorithm. The effect that changing the step size has on the computation time is showed in section 9.1.3.

The output of the algorithm is a collection of solutions that fits the design preferences. These solutions have diverse layouts and performances. It is quite often the case that a choice will have to be made between the layout and the performance. For a certain roof, some of the solutions are shown in figure 9.4.

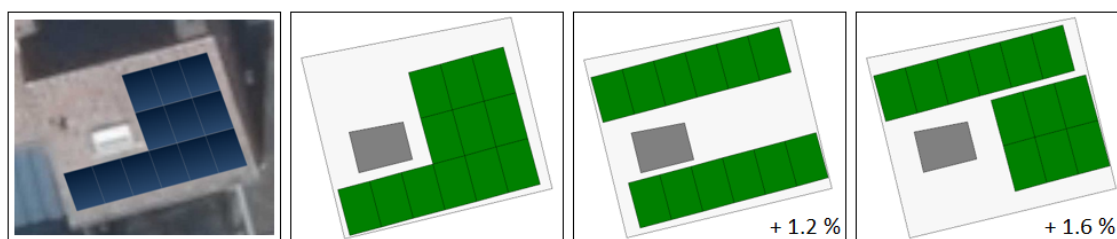


Figure 9.4: Different solutions for a roof with 12 panels to place. The performance difference compared with the manual solution is indicated.

The performance of the solutions differ from each other. There are a lot more solutions that are aesthetically unattractive and there is a high chance that one of those solutions will have a better yield than an attractive looking solution. There is a trade off between aesthetics and performance, which was also mentioned with the layout grading in chapter 6. The algorithm can present different options and then a human can choose one of them.

Flat roofs

This algorithm was not tested on many flat roofs to compare its performance. This was due to the effort it takes to collect the roofs and the fact that the layout grading for a flat roof was not completed yet. The algorithm was instead tested on a few flat roofs, just to test if it works in principle. The flat roof that was used for this is also used to test the maximum panel placement algorithm. It could however also serve as an example roof to demonstrate this algorithm. The algorithm was run for the amount of panels that were placed on the roof. It was noticed that a solution was not found when the panels were faced south direction. The panels were therefore orientated in direction of the roof. The original layout and three of the automatic layouts can be seen in figure 9.5.

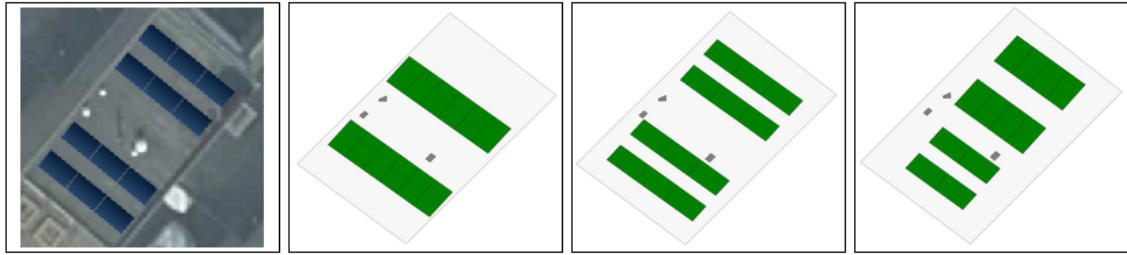


Figure 9.5: Three solutions for a flat roof with 12 panels.

The algorithm presents many different possibilities for placing 12 panels in this orientation. Only a few were shown. As was said earlier, the layout grading is however not yet adapted for flat roofs. This means that portrait is still preferred, while this might not necessarily be the case for flat roofs. It can also be seen that there seem to fit more panels on the drawn roofs than on the roof on the photo, this is due to the inaccuracy of drawing the roof polygon. The panels are the exact same size, but the roof is that is drawn is a bit larger.

Another flat roof was also tested and the amount of panels to be placed was set to half of what was originally placed on the seemingly maximally filled roof by the designer. The panels were rotated to face south. The algorithm produces a wide range of possible solutions. Some of those are presented in figure 9.6.



Figure 9.6: Some solutions to a flat roof when placing a limited amount of panels with a south orientation.

The possible solutions will have different performances and of course different layouts. The layout will be less important as panels on flat roofs aren't as visible as those on pitched roofs. The algorithm will however find the solutions that have the highest predicted yield. The algorithm can use a combination of landscape and portrait orientated panels or only a single orientation. The necessary row distances are not violated. These row distances could also be calculated when entering the system characteristics. Not only can the orientation be freely chosen, it is also possible to chose an east-west orientation. Some results for the east-west orientation is shown in figure 9.7.

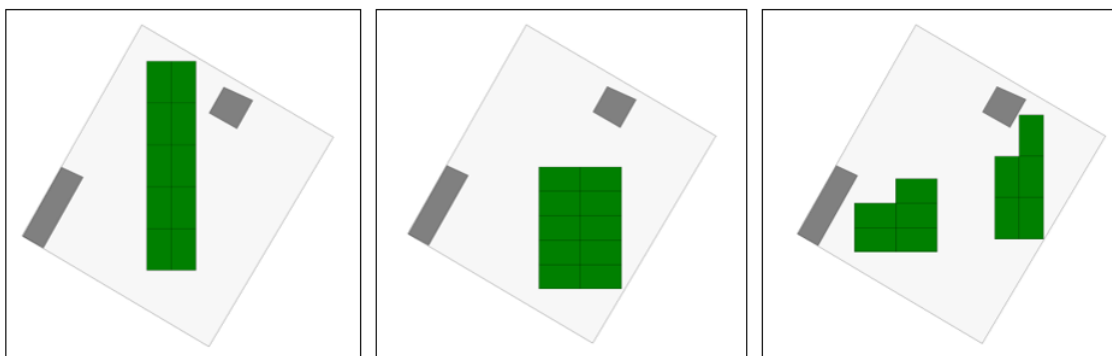


Figure 9.7: Some solutions to a flat roof when placing a finite amount of panels, in an east-west orientation.

Sometimes an obstacle will cause a panel to be removed, as can be seen in the third solution. This can result in more panels facing west than facing east. Such situations can be included in the grading of the layout.

9.1.2. Differences in the performance between the designs

The predicted performance of the automatically designed solutions and the manually designed solutions varied quite a lot. The differences are visualized the figure 9.8.

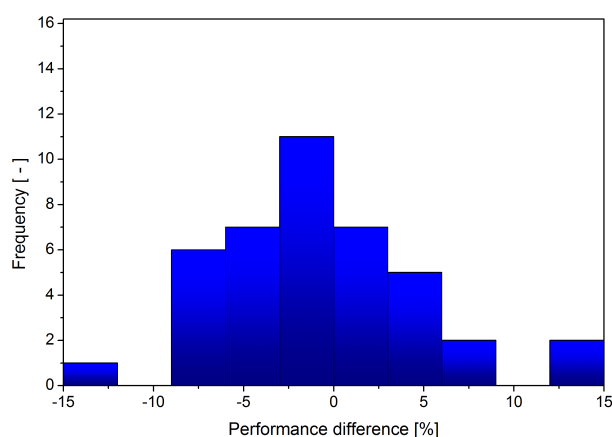


Figure 9.8: The performance differences with the manual designs for the pitched roofs with a finite amount of panels.

Such differences might be expected for totally different layouts, but were also the case for almost similar layouts. Sometimes the automatic design and the manual design were nearly identical and had still a big difference in performance. An example of this can be seen in figure 9.9.

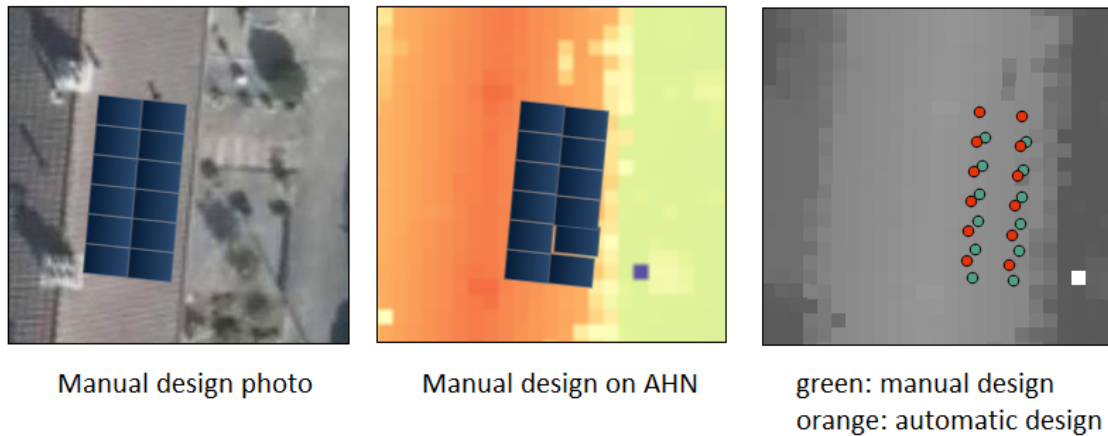


Figure 9.9: An example of very similar designs, but with a performance difference of almost 14%.

The reason for such a difference in performance can be explained by the fact that a single bad performance value affects the entire string. This happens, because the string is connected in series. Sometimes the AHN data has some bad pixels and this will result in a performance that deviates from what would be expected on that spot. If one panel ends up in one of those bad pixels it will reduce the performance of the total string. Looking at the exact positions of the panels, the performance should not differ as much as it does.

There is also another factor that contributes to differences in performances. The manual solution sometimes looks good on the photo, but that is misleading as it really only matters how the panels lie on the AHN layer. Some of them were really bad and were corrected by making a new, but similar design that would lie correctly on the AHN layer. An example of this can be seen in figure 9.10.

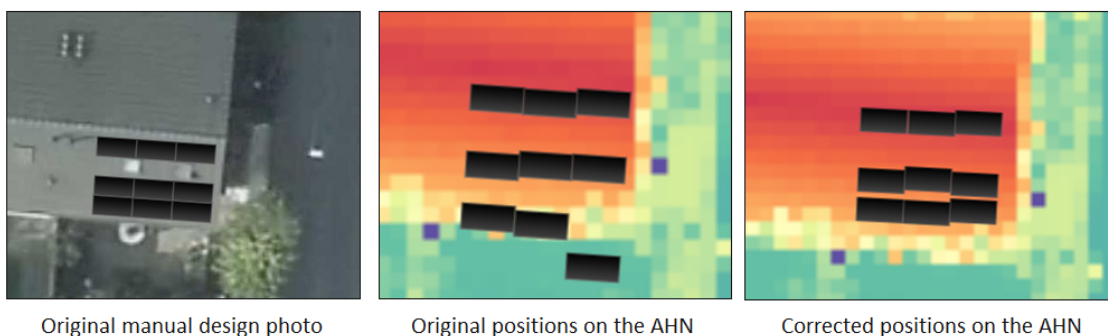


Figure 9.10: The correction of a manual design, due to the bad placement on the AHN layer.

In total about 68% of all the systems had a difference in performance of $\pm 5\%$. The average when taking all the absolute differences is 4.02% with a sample standard deviation of 3.42% and when just taking the normal average it is -0.83% with a sample standard deviation of 5.26%. Due to the differences in unexpected scenarios and the small sample size, not much can yet be said about whether the algorithm will find a better position on the roof than a manual designer would. The results do however indicate that the highest performing options are found on the performance grid that is used in the algorithm.

9.1.3. Algorithm duration compared to manual designing

The time that the algorithm takes depends on several factors. This makes answering this question not straightforward. The following aspects have an effect on the algorithm computation time:

- The algorithm settings
- The design requirements

- The size of the roof and whether it has obstacles
- Whether it is a flat roof or not

The data gathered by all the testing can give an indication for how long the algorithm will take in certain circumstances. The computation were run once on a server of Solar Monkey. It was about four times faster than the HP Elitebook 8540w. The exact specifications is however not relevant, as it is about order of magnitude. That is also the reason that the test were not run multiple times, which would be required to get accurate calculation time.

The algorithm consists of three steps and the computation time was measured for each part. The total computation time for each roof was determined and are gathered in the plot in figure 9.11.

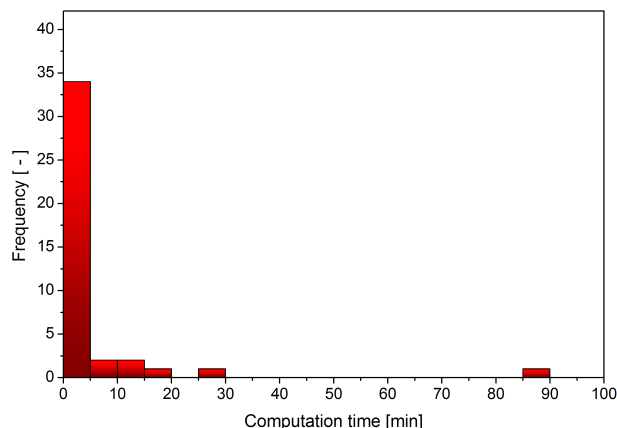


Figure 9.11: The computation times of the pitched roof with a finite amount of panels.

One roof had a very long total computation time of around 88 minutes. The reason for this is the large roof and the required 12 panels enable a lot of possible combinations.

All of the other test runs were completed within 29 minutes and about 88% was done within 10 minutes. The average duration was 5.8 minutes with a sample standard deviation of 14.2 minutes. The average is however influenced greatly by the large computation of 88 minutes, which also explains the large standard deviation. Dropping this large computation changes the values to an average of 3.8 minutes with a sample standard deviation of 5.5 minutes. The combined solutions step of the algorithm requires by far the most time, the trivial and large solutions steps each take only about 3.3 and 4.7 seconds on average respectively.

It is also observed that the trivial solution step duration really depends on the roof shape and the amount of panels to be placed. Sometimes there is no panel section that fits and the algorithm is done within a fraction of a second. The large solutions step requires about the same time for each roof. This is because it always places large sections, regardless of the roof size. The only deviation is caused by the amount of panels that have to be checked to see if they fit.

Some test were run to get a feeling for how the complexity of the situation scales with the computation time. A simple rectangular roof without obstacles and an increasing number of desired panels was tested. The surface area of the roof was about $35m^2$ and the panels were 165 cm by 100 cm in size. The desired amount of panels for the design was increased from 4 to 12 and the average time of three runs was recorded. The computation times for each part of the algorithm can be seen in figure 9.12.

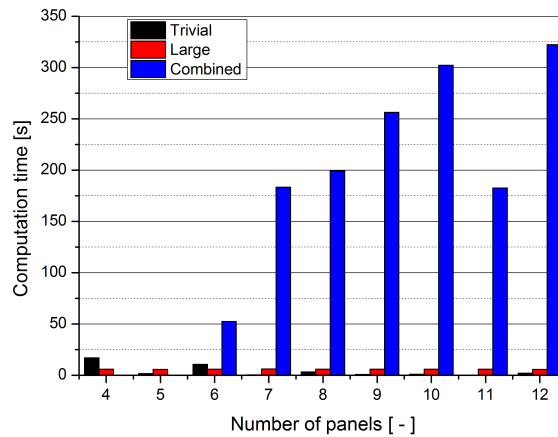


Figure 9.12: The duration of the algorithm when the amount of panels is increased.

For only four or five panels the algorithm spends almost no time on finding solutions for two sections, this is because there are no combinations to be made that will result in a solution. With more panels the third part of the algorithm consumes a big majority of the computation time. The computation time depends on the amount of possible solutions that are checked. Sometimes if there are more panels, the amount of combinations will drop. This is then because there are less ways of making a panel section for that amount of panels or that the panel section become so large that they don't fit on the roof. In general however, more panels mean more possible combinations to go through and thus a longer computation time.

In a second test on the same $35m^2$ roof, the step size was varied from 10 cm to 100 cm and the amount of desired panels was kept fixed at 8 panels. The total computation time for each scenario can be seen in table 9.1.

Table 9.1: Computation times with different step sizes.

Step size [cm]	10	20	30	40	50	60	70	80	90	100
Computation time [min]	193	11.7	2.3	0.65	0.35	0.22	0.14	0.13	0.11	0.11

It becomes clear from table 9.1 that decreasing the step size will increase the computation time dramatically once the size is reduced below 20 cm. A smaller step size means more positions and that means more possible combinations of sections to be made. The amount of positions will quadruple if the step size is cut in half. Every panel section will be placed in all of the positions and combinations will be made will all those sections, so having a lot more positions will rapidly increase the number of combinations.

A third aspect that influences the computation time is the presence of obstacles. A small obstacle such as a chimney in the corner will create extra panel sections. Sometimes the algorithm becomes inefficient if there is a certain obstacle. An obstacle will always prompt extra panel sections, while that might not always be required. Two different panel scenario's are depicted in figure 9.13, with both having an obstacle that prompts extra panel sections.

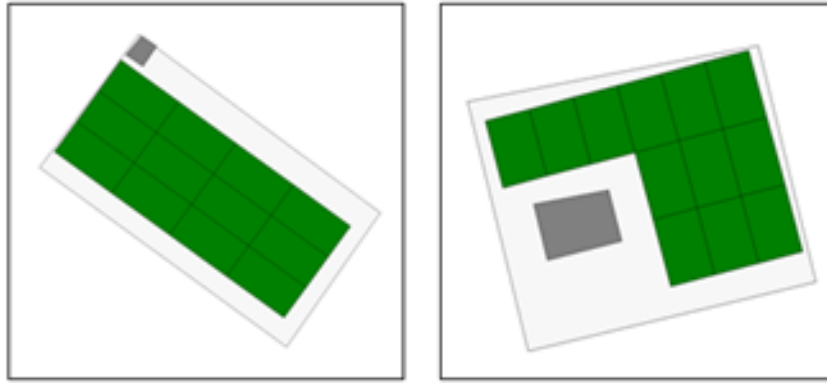


Figure 9.13: Two different roof scenario's that prompt extra panel sections due to an obstacle.

The left roof in figure 9.13 has an obstacle in such a position that the extra panel sections will not help much in the search for a suitable solution. The solution will almost always be trivial and if the roof has to be filled to a maximum, the grid shifting method will be applied. The roof on the right however will require those extra panel sections to find more solutions. The algorithm does not recognize the differences between these two scenario's. For further improvement, the algorithm could perhaps also look at the kind of obstacles on the roof and adjust accordingly.

A flat roof takes longer to compute than a pitched roof. This is due to the added computations that happen to take care of the row distance. The addition of the panel edges take some extra time.

9.2. Maximal panel placement algorithm for pitched and flat roofs

A total of 15 flat roof scenario's were used to test this algorithm. The orientation of the panels was calculated by the algorithm and was chosen to be in line with the orientation of the roof. This is also the orientation that was chosen by the manual designer.

9.2.1. Azimuth differences algorithm and manual designs

For flat roofs the differences between the orientation chosen by the designer and the orientation that was calculated by the algorithm were determined. For the 15 roofs the results can be seen in the graph in figure 9.14.

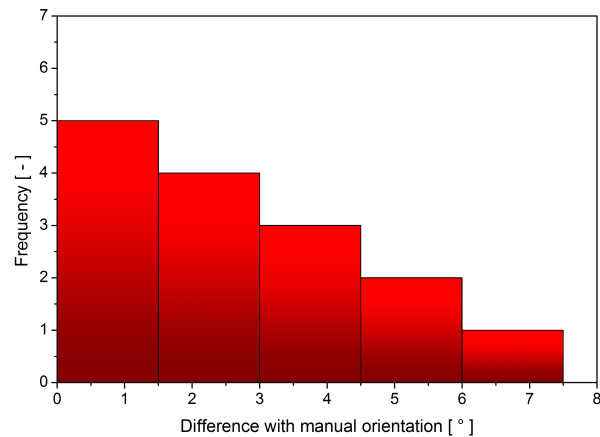


Figure 9.14: The absolute differences in the orientation determined by the algorithm and the one chosen in the manual design.

Sometimes the calculated angle is larger and sometimes the calculated angle is smaller than the orientation that the designer had chosen. Looking at the absolute differences it can be seen that differences occur less frequent the larger they become. The biggest difference in the tested sample was 7.4 degrees. When taking the absolute differences, the average difference is 2.65 degrees with a sample standard deviation of 2.05 degrees. These results make the current method for determining the orientation of the panels along the orientation of the roof a viable option.

9.2.2. Differences in the amount of panels

As was mentioned earlier, the intention of the person designing the system that the algorithm is compared with is unknown. Sometimes this leads to uncertainty about whether the intention was to fill the roof to a maximum or not. As an example the same roof is used as was used in figure 9.5 and can be seen in figure 9.15.

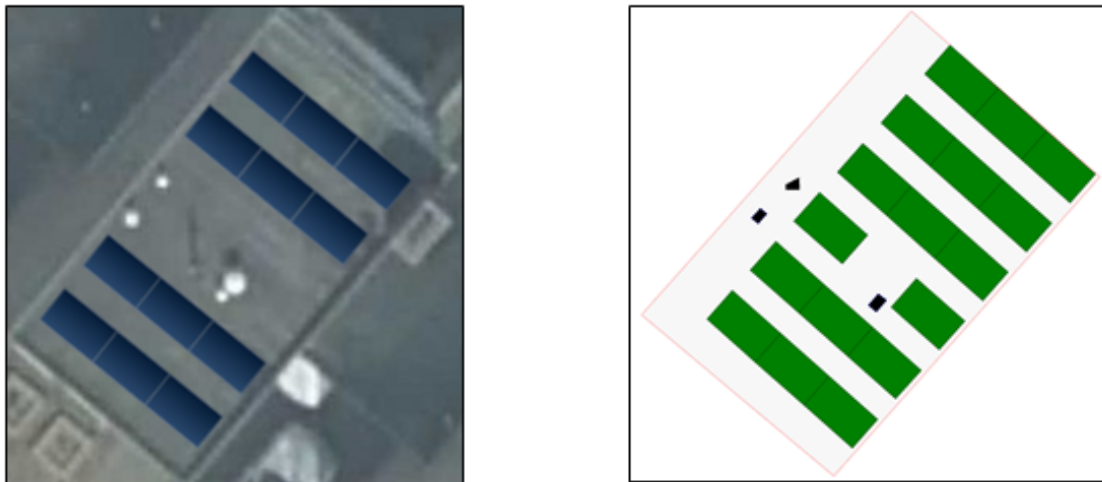


Figure 9.15: A manual and automatic design for a flat roof PV system.

In the example shown in figure 9.15, the algorithm determines that more panels fit on the roof than were placed in the manual design. This is however also to be expected when looking at the manual design. With some shifting of the panel grid, more panels would also have been placed manually. This was perhaps not done, because no more panels were desired. This roof has now been tested for a

maximum amount and shows that more can fit that were placed by the designer and has been run for the same amount of panels as the designer and showed that a lot of variations are possible.

Situations like in figure 9.15 make it difficult to determine how often the algorithm will place more panels on the roof than was manually done. It would require an interpretation about the intention of the designer. The inaccuracy with the drawing of the roof polygons also complicates this. It has to be made sure that the dimensions of the panels and the roof are the same for the algorithm as for the manual design.

This could not yet be guaranteed and for that reason not a large sample of roofs is tested to see if the algorithm can fit more panels than the manual designer could. The results of a few scenario's are however analyzed to determine whether the algorithm is expected to return the maximum amount of panels. Two manually designed pitched roof systems and the maximum panel placement algorithm results are shown in figure 9.16.

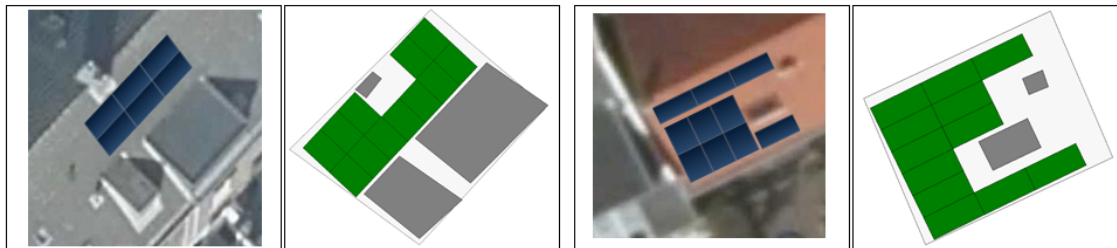


Figure 9.16: Two manual designs on pitched roofs and a maximum panel placement algorithm result.

The algorithm seems to find the maximum amount of panels for a single large section. The designer for the left roof probably didn't mean to fill the roof to a maximum, but the algorithm does show a neat solution for if that would have been the goal. With the design on the right it seems like the designer wanted to put as many panels on the roof and was doing that by building multiple sections, running the maximum panel placement algorithm will find a nicer looking possibility within seconds. Results like this give a sense of what the algorithm can do. More testing has to be done if the roof and panel dimensions can be extracted more precise.

One thing that the algorithm can do quickly for flat roofs is change the orientation of the panels and calculate how many will fit. The row distance might have to be adjusted when the orientation changes, but in principle it can be easy to see if adjusting the orientation slightly will increase the amount of panels that fit. This is done for the very large roof that can be seen on the right side of figure 9.19. The azimuth in line with the roof was calculated to be 186 degrees, but the algorithm was also run for three smaller and three larger azimuths. The azimuths increased or decreased with steps of 3 degrees. The total amount of panels that fit per panel azimuth can be seen in figure 9.17.

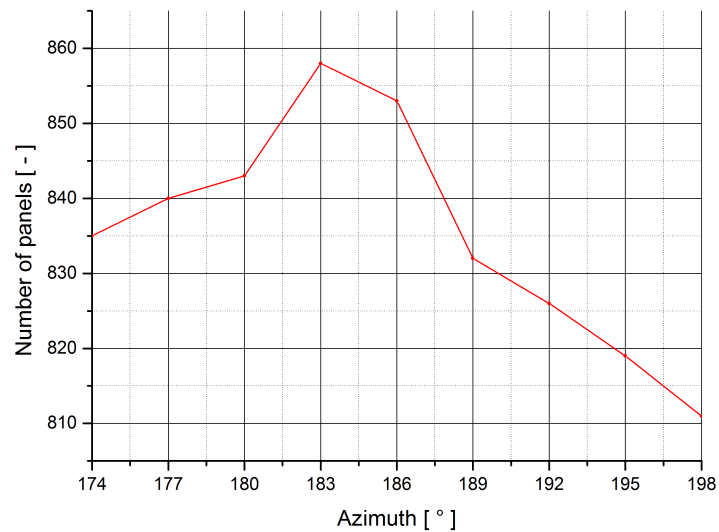


Figure 9.17: The total amount of panels that fit for different panel azimuths. The row distance is kept the same for simplicity.

It turns out that more panels fit on the roof if the azimuth is slightly smaller than was calculated. The calculated azimuth was 186 degrees, but more panels fit with an azimuth of 183 degrees. If the relationship between the optimal row distance and the azimuth is known it can be an extra optimization to not just run the algorithm for the azimuth calculated, but also for azimuths slightly smaller or larger. The obstacles on the roof will have an influence on which azimuth will be best to fit the most amount of panels.

9.2.3. Algorithm duration compared to manual designing

The algorithm for the placement of maximum amount of panels takes only a few seconds. There are two big factors that influence the duration of the algorithm. The first one is the step size of the grid shift. This was set to 10 cm for the tests that were done. The second factor is the size of the roof. Another factor would be the size of the panel, since smaller panels mean more panels, but that is only a small factor. The calculation of the value for each panel was not done, only the maximum amount of panels was searched. The duration of the algorithms were plotted against the area of the roof and this graph can be seen in figure 9.18.

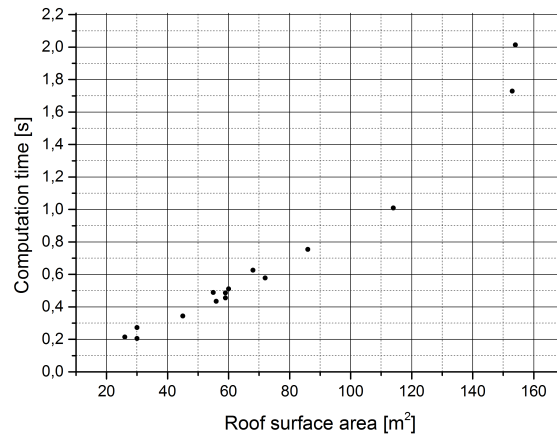


Figure 9.18: The computation times for flat roofs with different surface areas.

The relationship between the area and the computation time seems linear from this small sample of roofs. Three larger roofs were also measured, but displaying those in the same graph would however distort the graph a lot, because their surface area is so much larger. The surfaces areas were 3311 m², 1974 m² and 551 m² with computation times of 249 s, 70 s and 5.9 s respectively. Since the duration scales with the surface area it makes sense to look at the computation speed in terms of area per second.

The average time per square meter is 104 m²s⁻¹ with a standard deviation of 35 s. It matters a lot whether the roof has obstacles in it. In the current algorithm each panel is checked for overlap with the obstacles. The large roofs have a lot of obstacles and this affects the average computation. The average computation time for roofs without obstacles is 121 m²s⁻¹ with a standard deviation of 16 s. The computation is faster without obstacles and the standard deviation is smaller, indicating that a larger amount of obstacles increases the computation time.

Just looking at the numbers does not immediately give a good idea of how fast the algorithm is. In figure 9.19 there are three flat roofs and the corresponding computation times, to get a feeling for the speed of the computation.

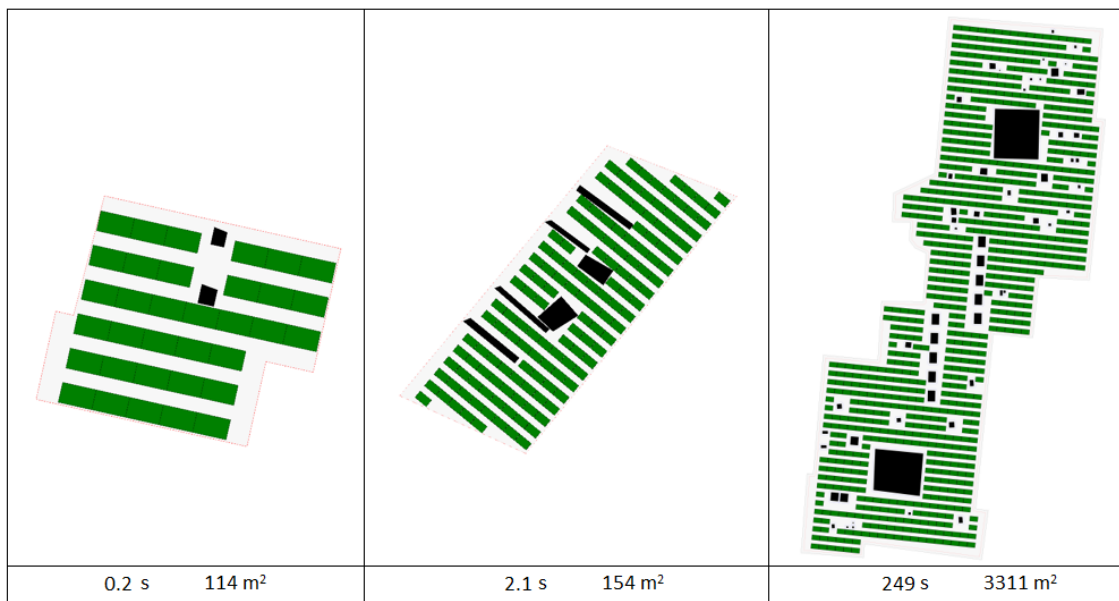


Figure 9.19: Three flat roof surfaces and their computation time.

The relation between the size of the roof and the duration is caused by the fact that a larger panel grid has to be created and each panel has to be checked whether it fits on the roof. Designing these system by hand is however time consuming. The orientation has to be determined and then the panels have to be placed in such a way that the most amount of them will fit on the roof. Even if the grid can be created fast by dragging and dropping, the exact position will take several tries. The panels that don't fit will have to be removed manually and for a large roof this could perhaps take up to 15 minutes and will most likely be less accurate than the algorithm.

9.3. Finite panel placement algorithm for a panel range

The finite panel placement algorithm was used on a pitched roof and it looked for solutions with either 8, 9 or 10 panels. This produces a lot more results than looking for only one amount of panels. The different solutions that are generated are discussed, as well as the time it takes when looking with range of panels.

9.3.1. Diversity among the produced solutions

The algorithm produces a lot of solutions and these are all ranked according to the multi objective optimization explained in chapter 7. Some of the solutions are taken and used to discuss the diversity of layouts and performances that are presented. A portion of the produced results, including their fitness, can be seen in figure 9.20. It should be noted that these results are top views of a roof with one window and the roof is facing west. This means that the top of the roof is at the right vertical edge.

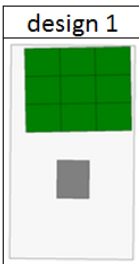
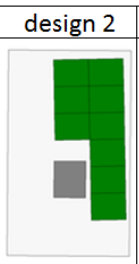
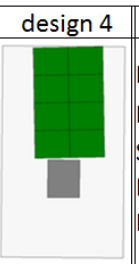
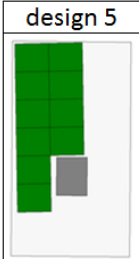
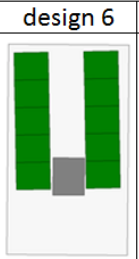
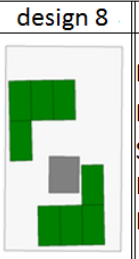
	design 1 fitness 1 P _{total} : 2453 P _{avg} : 272 S: 1 H: 1 L: 810		design 2 fitness 2 P _{total} : 2468 P _{avg} : 274 S: 1 H: 0.86 L: 810		design 3 fitness 3 P _{total} : 2696 P _{avg} : 269 S: 1 H: 1 L: 1350		design 4 fitness 4 P _{total} : 2188 P _{avg} : 274 S: 1 H: 1 L: 720
	design 5 fitness 5 P _{total} : 2683 P _{avg} : 268 S: 1 H: 0.91 L: 900		design 6 fitness 6 P _{total} : 2727 P _{avg} : 273 S: 2 H: 1 L: 900		design 7 fitness 7 P _{total} : 2725 P _{avg} : 273 S: 1 H: 0.91 L: 900		design 8 fitness 8 P _{total} : 2117 P _{avg} : 265 S: 2 H: 0.8 L: 1080

Figure 9.20: Different solutions for a panel range of 8, 9, 10.

Eight different solutions are presented and the range between 8 and 10 panels. Some of them are objectively better than other according to the ranking system, but it was chosen to display a wide variety. It already becomes clear that certain choices can be made between the presented solutions.

The highest performance will come with solutions that consist of 10 panels. Those solutions however will generally have a lower average performance per panel. For the highest total performance, design 6 has to be chosen. It also scores high in the H factor and the average panel performance. This design however consists of two sections and that might not be preferable when it comes to installation. Design 7 might then be a better option with 10 panels.

A solution that doesn't have a high total, but is still a good option is design 4. It scores the highest in all grading criteria, except the total performance. If the total amount is not very important, but aesthetics and value is then this is a very good option.

In the end it will be up to the user of the software to pick a layout out of several options. How many that should be presented can easily be changed in the code. For large scale implementation the algorithm will probably have to pick just a single solution. It can then be predetermined which of the grading criteria is more important.

9.3.2. Algorithm duration compared to manual designing

The average duration over three runs for this design preference is 217 seconds. this can be compared to the computation for only one of the panel totals. Table 9.2 shows the computations for the entire panel range and for the panel totals separately, each time being the average of three runs.

Table 9.2: Calculation times for different design preferences.

Design preference (number of panels)	8	9	10	8, 9, 10
Computation time (s)	116	129	118	217

The duration increases with the amount of panel totals that is looked for. It is however faster to compute the algorithm with a panel range than to compute every panel total separately. The reason for this is that certain computations will be done multiple times if they are calculated separately. Step 2 of the algorithm is almost the same every time, except that it allows solutions with a different amount of panels. The determination of roof points and the placement of panel sections has a lot of overlap, certain sections have to be placed for all panel totals. This means that certain panel sections are evaluated multiple times if the panel totals are calculated separately. There are probably more smart tricks that can be used when running the algorithm with a panel range that could reduce the computation time. It is a vital part of the final algorithm, as running for a panel range will then probably become the standard.

9.4. Results summary

In this section the results of the validation is summarized in three tables. Table 9.3 shows the amount of times the manual design and/or alternative solutions were found with the finite panel placement algorithm for a pitched roof.

Table 9.3: Solution finding rates for the finite panel placement algorithm for 41 pitched roofs.

Panel adjusted	Manual design found [%]	Alternative design found [%]
false	61	83
true	80	100

The fact that at least alternative designs are found in every scenario, if the panels are adjusted, gives good hopes for the possible implementation of the algorithm in the future.

Table 9.4 shows more data about the 41 pitched roof scenario's.

Table 9.4: Results of the finite panel placement algorithm for 41 pitched roofs.

Data	Average	Sample standard deviation
performance difference [%]	-0.83	5.26
performance difference absolute [%]	4.02	3.42
Algorithm duration [min]	3.8	5.5
Roof size [m^2]	37.2	13.3
Number of panels [-]	9.8	2.4

Table 9.5 shows information about the maximal panel placement algorithm for flat roofs, tested on 15 scenario's.

Table 9.5: Results from the maximum panel placement algorithm for 15 flat roofs.

Data	Average	Sample standard deviation
Azimuth difference with manual [$^{\circ}$]	2.7	2.1
Algorithm speed per area, with obstacles [m^2s^{-1}]	104.5	34.7
Algorithm speed per area, without obstacles [m^2s^{-1}]	120.7	16.4

The azimuth determination works wells. The algorithm speed differs for roof with obstacles. The speed is however with or without obstacles still significantly faster than making the design manually.

10

Conclusions and recommendations

10.1. Conclusions

In this thesis the project for the development of an automatic PV system design algorithm was started. The project was under supervision of both the TU Delft and the company Solar Monkey. The development of the algorithm was explored and several different algorithms were made and tested during this thesis. In this chapter conclusions are drawn about some of the algorithms that were developed and tested. Their feasibility and possible future improvement is discussed. At the end of this chapter there are recommendations about the improvement of already developed algorithms and about the next steps that will have to be taken in the process of creating a complete automatic PV system design algorithm.

10.1.1. Panel placement algorithms

The panel placement algorithms fall in two broad categories. One that completely fills the roof and one places a finite amount of panels based on a certain design preference. Both of these categories were tested for both pitched and flat roofs.

Non maximum filling algorithms

The non maximum filling algorithm was tested for the panel layouts it produced, the performance of the layout and the duration of the computation.

I) PV system layouts

The panel layouts were evaluated based on whether the same layout was produced as the installer and whether reasonable alternatives were presented. The main factor that determined this outcome was whether the shape and size of the polygons that were drawn were correct. There is an inaccuracy of about half a meter due to the AHN resolution and the positioning of obstacles on the roof had to be estimated. This played a big role. The other reason is the positioning of the panel sections. The panel sections were only placed at certain positions and sometimes this positions was not quite right to make a panel section fit, even though it could fit on the roof. The bad influence of these factors can be reduced and these factors will be addressed in the recommendations. The conclusion is that the algorithm will almost always find the layout made by the installer and provide alternative solutions if those are possible, if the roof and panel dimensions are correct.

II) PV system performance

The performance differences between the manual made and the automatically designed systems proved to be higher and more unpredictable than expected. The reasons for this are the series connection of the panels, the sometimes big variation in AHN data point values and the sometimes poorly placed

panels on the AHN by the installer. The most optimal position was found in the performance grid, but that didn't show up in the results when the layouts were calculated with the Solar Monkey software. The conclusion is that the algorithm will find the optimal position for the performance grid it works with. In order to have it work well in practise, this performance grid has to be accurate.

Another thing to note is that in the algorithm the combined solutions are treated as two separate strings. When the panels are calculated with the Solar Monkey software, they are all treated as a single string. This doesn't matter for the comparison, but it does make a difference in finding the best position on the roof. If all the panels, even of different sections, are treated as a single string in the algorithm it would produce some different results. These differences should be analyzed in the future to determine the best way of running the algorithm.

III) Algorithm duration

The duration of the algorithm is very dependent on the specific roof and the design preferences. More computation time is required for a larger roof, for more panels and in the presence of obstacles on the roof. The algorithm consist of three parts and the last part takes significantly more time. This is the part where solutions of two sections are created. Often there are many combinations possible, which makes this a long process. With an average computation time under 10 minutes it is however a computation time that is not too long, when compared to manual design. Especially with the outlook of speed improvements in the future.

Maximum filling algorithms

Due to the inaccuracy of the roof dimensions and the uncertainty of the intentions of the manual designs it was not possible to test whether the algorithm would find a solution with more panels for a reasonable sample size.

I) Azimuth determination

The method to determine the orientation of a flat roof works well and get the orientation of the roof within a few degrees. As an extra optimization several different panel orientations near the calculated orientation can be tried to try and find an even better orientation.

II) The amount of panels placed

The algorithms for flat and pitched roofs were tested on about 15 roofs of each type and the results indicate that the maximum amount of panels in a single orientation will be found. There is however one parameter that has big influence on the accuracy and that is the distance that the grid of panels is shifted each iteration. A smaller distance will result in finding the optimal placement of the grid with a higher certainty, but it will require a longer computation time.

10.1.2. Inverter algorithms

Algorithms were developed that could predict the miss match between panels in order to determine the losses that occur when connecting them in series. With these algorithms the next step was taken to dry to create a method that would divide a group of panel in separate strings in which the total performance would be optimal. Lastly a previous research was used to determine the optimal size of an inverter based on some installation characteristics.

Panel miss match

Two different methods were developed to determine the miss match between two panels and between a group of panels. Both methods produced similar miss matches for the same scenario's. They will however provide slightly different miss matched, because the methods use a different calculation. The first method is based on stacking obstacle views, which is a fast calculation and the second method is based on the movement of the sun across the sky throughout the day and takes a bit more time. Both methods were however not yet validated with real data. This would require a lot of performance data of installed PV systems and the individual panels in the system. A method to setup a validation

experiment is discussed in the recommendations.

String building

Even though validation was not yet done, the first steps were however already taken to use these methods to divide a group of panels into separate strings. The method determine all possible valid string configurations and then does a calculation for all these configurations. A valid string configuration consists of strings in which all the panels are connected in a single block, in which diagonal connections are not counted as a valid connection. This method works reasonable well on a small scale. There is however a problem when it is used for large systems. Determining all the valid string configurations becomes problematic due to all the possible panel combinations.

One idea was tested in which the large panel section is first split into separate smaller sections. Then each of those separate sections is solved. This method does seem to work. A question however rises and that is how to divide the panels into sections. Another idea is that the problem can perhaps be solved with graph theory. The panels will all be represented as nodes. The miss match between adjacent panels can be calculated and then a weighted graph can be obtained. A path through the nodes can be seen as a valid string. How these miss matched can be used to find the optimal division of strings is not yet explored.

Inverter sizing

The inverter algorithm for the size was based on the thesis by Jessica and the algorithm provides an inverter size range that corresponds to the optimization done in her thesis. This algorithm can be used to quickly get a sizing for the inverter, but in a more complete algorithm the code used by Jessica should be implemented, instead of interpolating from a table.

10.2. Recommendations

This part of the chapter is about the recommendations for the developed algorithms, suggestions for the rest of the project and other ideas for the future. First some recommendations are given for the already developed algorithms, secondly there are some recommendations about what could be done next in the development of automatic PV system design algorithms, thirdly some suggestions are given for the large scale implementation and lastly the short term implementation of the algorithms for single roofs is discussed. The latter being a point of interest for Solar Monkey.

10.2.1. Current algorithm improvement and optimization

A lot of different algorithms have been tried during this thesis. Almost all of them have been adjusted during the development stage. There is however quite some room left for further improvement and perhaps different approaches to the way things were done. This section consists of changes that can be made to the current algorithm. Other features than could or should be added are discussed in a later section.

Maximum filling algorithms

The maximum filling algorithms rest on the concept of shifting a grid with the size of the panels across the roof in several steps. The step size with which this is done will depend on the speed of the algorithm in its final form. A smaller step size will increase the chance of finding the exact best grid position, but it will require more computations. The general idea behind this concept seems pretty solid, improvements can however be made in the way that it is coded. Someone with a good knowledge of this can likely find some way to optimize the code. The concept can however also be expanded a bit by allowing additional shifts of the panels. The only shifts that happen right now are shifts of the entire grid, but individual rows could also be shifted. This is shown in figure 10.1.

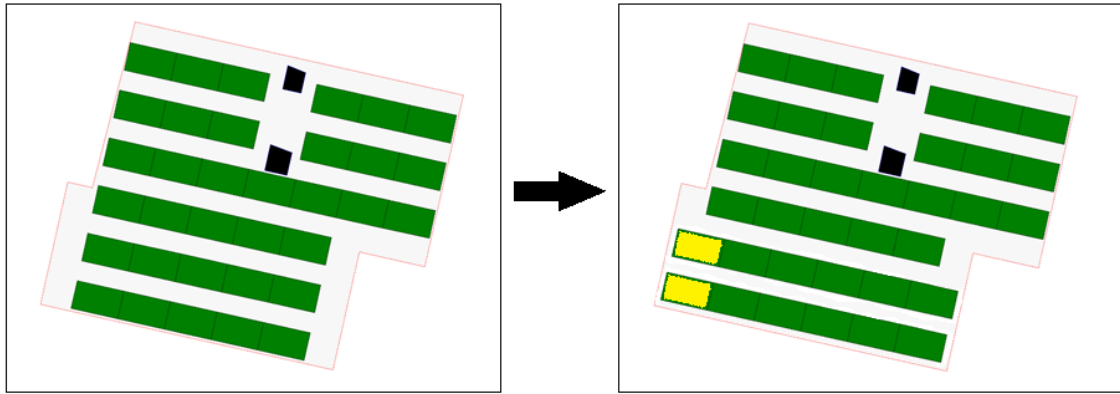


Figure 10.1: Example of a roof where extra shifts result in extra panels.

It should however be investigated whether installers like such a layout, as the rows are no longer symmetrical. An additional feature yet to be implemented is the option to stay above a certain ROI. The easy way to do this is by removing the panels that perform below a certain level. A more complex method would be to calculate the ROI at each grid shift and then to pick the best one. These concepts are also relevant for pitched roofs.

Non maximum filling algorithms

The non maximum filling algorithm is more complex than the maximum filling algorithm. It consists of several stages and has more parameters that can be adjusted. The general approach of the algorithm is to build solutions by combining different optimally placed panel sections. Suggestions will be made for all of these steps in the algorithm.

• Step: Determine panel sections

I) Choose panel sections based on speed and experience

An important aspect to determine is which panel sections are going to be used to build the solutions. It is obvious that very large sections should not be tried for small roofs. The list of panel section is a function of the roof size, the presence of obstacles and the design preference. It also depends on how fast the computation will be in its final form. Extra panel section will only improve the chance of finding a good solution, but it will require more computations. To optimize this it is probably best to start out with a lot of different panel sections to have a large chance of creating a good solution. This panel section selection can then later be scaled down if necessary after the speed optimizations.

II) Use non rectangular panel sections

Another thing that might be worth investigating is the use of panel sections that are not rectangular. Especially for triangle shaped roofs a non rectangular section can be interesting.

• Step: Determine roof positions

I) Roof positions based on algorithm speed and accuracy requirement

The positions on the roof is a trade off between calculation speed and the chance of finding the exact optimum solution. More positions means a higher chance of finding a solution, but it will take more time. Once the speed optimizations of the algorithm are implemented it can be debated how the panel positions should be. This can be determined by the desired overall calculation time and experienced success rate with certain settings.

II) Use only good positions

Another thing that can be explored is to only use roof positions that have a performance value above a certain threshold. For a scenario where the roof is very large compared to the amount of panels to be placed, this can be beneficial. It can reduce the computation time, by removing the bad roof

positions as possible positions of panel sections. This approach was initially also used, but later abandoned.

III) Reintroduce local searches

It could also be possible for large roofs to use a step size that is relatively large and to reintroduce the local search again. A local search will then be performed for the good panel sections. What this method then effectively does is first broadly search for the good areas on the roof and then fine tune the exact positions of the panel sections. The step size should not be simply fixed, but depend on the roof surface.

• **Step: Build solutions**

I) Don't use bad sections

The building of solutions that consumes the most time is when multiple panel sections are combined. In the current algorithm this is done quite straightforward. All of the panel sections used in this part of the algorithm are combined together and then the results are filtered and ranked. This means that also very bad sections are combined with every other section, while it can already be known that such a bad section will never combine with something to make a good solution. This means that unnecessary combinations are made that just increase the computation time. Perhaps only panel sections that perform above a certain threshold should be used in combinations. There are also panel sections that are only left with one or two panels due to overlap. These sections are still combined, but a solution with one large section and one section with only one panel will never be chosen. Such tiny panel sections could be left out of the combinations to speed up the algorithm.

II) Allow overlap of sections

The algorithm could expand the search to also include combinations in which the panel sections are allowed to have overlap. This was not included, because it would require an extra calculation of the fitness that would slow down the algorithm. With more efficient coding this can be however be implemented.

III) All solutions with more than two sections

Expand the search for solutions that consists of more than two panel sections. These are often not required, but for large roofs it could be interesting. The selection of panel sections to use should then also be adjusted to this.

• **Step: Evaluate and rank solutions**

I) Include system losses

The current evaluation of the solutions does not include any losses. The panels can be treated as a series connection or as parallel connection without any losses. To determine the right system configuration it will also be relevant to include the losses caused by the type of inverter that is chosen. Other system losses that depend on the layout could also be included in the future.

II) Improve filter mechanism

The filtering of duplicate solutions is not yet fully implemented correctly. There will be rare cases in which the filtering doesn't work. In some situations a panel sections is split into separate sections due to obstacles. There will then be still a single layout matrix with the center elements filled with zeros. There is a possibility that such a section can be equal to two separate sections with separate layout matrices. This will however not be identified as a duplication, because the first solution has a single layout matrix and the second solution has two layout matrices. In future development this single matrix should be split into separate matrices with both its own center point. Then this duplication will be recognized. This was not done yet, because it was only realized at the end. It is not very likely that such a situation will occur.

Inverter algorithms

I) Validate panel miss match

The inverter algorithms still lack a good form of validation. A predicted miss match between two panels is not yet confirmed by measurements. This would however not be too difficult to test. It would require the performance of separate PV panels within a system, as well as the performance of the combined system in a single string. The defined miss match can be easily compared to the measured miss match. This data could be obtained from already installed systems, or a few panels can be set up for an experiment. With an experiment the miss match prediction can be explored in more detail, by playing around with partial shading on panels. It could be used to confirm whether the miss match prediction based on the obstacle views is correct. If that is the case it can be used the design of new systems.

II) Experimentally compare different string configurations

The string building based on miss match can provide higher performances than simple string configurations without much thought behind it. In this thesis it was not investigated how much these differences can be. The developed methods could also not be validated with real performance data. They could only be validated with the performance prediction from Solar Monkey. The miss match prediction is however based on that prediction method, so this comparison is not really fair. A good method of validation would be to setup a panel section of for example 28 panels and to divide that into four strings of seven panels. The configuration of the strings should be changed and the output should be measured. This will give an overview of how different string configurations perform. These measurement can then be compared to the predicted performance of these strings. Especially the order from best to worst should be compared. This is valuable information that can help to determine a good method to divide the strings. This setup can be done outside or in a controlled environment inside. This controlled method could be with artificial lighting, controlled lamp positions and self made obstacles. It can provide more variety of testing environments and probably faster reliable data, because there are no fluctuating weather conditions. It is however not as easy and cheap as just placing a panel section somewhere outside.

10.2.2. Future steps for automatic PV system design

Several different ideas about how to move forward with the development are discussed here. Some of them are reasonable short term, while others might come into play further down the road.

Combining the different algorithms

Eventually all the algorithms for different roofs have to be combined. This process has already been started and some prototype exists. The pitch angle of the roof determines whether a flat or pitched roof algorithm should start. There are however several settings that need to be determined for a well working algorithm for all scenario's.

I) Efficient use of the different steps

The non maximum filling algorithm still has some settings that should be determined. In some instances the desired design might have such an amount of panels that the roof is nearly completely filled. It would then make not much sense to try and find a solution with the combined solutions step of the algorithm. In such a scenario's either the trivial and the large solutions steps are more suitable. Perhaps a max filling algorithm can even be used in those cases. The worst panels could be removed to get to the desired amount of panels if there are too many. Which steps to uses shall depend on the roof surface area, the amount of panels desired, the presence of obstacles and if possible also the shape of the roof. The combined solutions steps takes by far the most time and using this when it is not practical is something that should be avoided. The other steps are much faster and are not very time consuming to run in any case.

II) Consider the inverter in panel placement algorithm

Inverters come in different sizes. These steps between these sizes are however generally larger than

the power of a single panel. This means that it can sometimes be the case that adding an extra panel to a system will require the change to a much larger inverter. This can make the system relatively more expensive. It can therefore be sometimes a more cost effective decision to not add that extra panel that causes the necessity of a larger inverter. Combining the inverter size algorithm with the panel placement algorithm can lead to a combined optimization instead of optimizing one after the other. That can be an overall better optimization.

Using multiple roof sections

Up till now the only thing that was considered is a single roof section on which the panels could be placed. In reality a building that is going to have a PV system has several roof sections that can be suitable to place panels on.

I) Identify suitable roof surface

One of the first steps in the designing process would then be to determine which roof section is the most suitable to use. This can be done based on the orientation, pitched angle, available space and a performance grid. Perhaps multiple roof sections are suitable, which means a larger system is viable. If a larger system is not desired, then it can also be an option to divide the panels over multiple roof sections. This will require some adjustments in the algorithms. If the roof sections have different pitch angles the performance of the panels on the roofs will be different. It should however be possible to adjust the algorithm in such a way that it works on multiple roof polygons.

II) Recognize east-west option with two roof surfaces

Another feature that is related to this is the option to have an east-west configuration on a building with a pitched roof. Some buildings are oriented such that one roof section is facing east and another is facing south. The algorithm should recognize this and design an east-west configuration by using two roof surfaces.

Multiple panel types

So far in all the algorithms a single panel type was used. The dimensions of the panel could be entered and the algorithm would look for a solution for that particular panel. The designing process should however also include the consideration between different types of panels. The roof can be shaded in such a way that some panels will have to be placed in a very bad position on the roof. In that scenario it might be a good option to go for fewer, but more expensive panels. Different panel types will also differ in dimensions. It can very well be the case that a certain panel type will not have the right dimensions for a neat solution, but another panel type does. The algorithm could look for solutions with different panel types and present the best options.

10.2.3. Large scale implementation

One of the long term goals of this project was to be able to automatically design PV systems for many buildings. That future is however still some steps away from the current status of the algorithms. There are still several things that needs to be developed for this goal to become a reality. Most of those things were never within the scope of this project, but their importance and difficulty have come forward some more. There are more aspects that need to be optimized, such as the speed of the algorithm and the parameters. Those are however also necessary for the shorter term small scale implementation and those are therefore discussed in that section.

Roof polygons extraction

The first important thing that needs to happen is the automatic extraction of roof polygons. This was currently done by hand, but in the future the algorithm should be able to do that with just the address as an input. The roof surfaces might not be a problem, because the AHN data can be used for that. The real challenge lies with the obstacles that do not show up on the AHN data. These are the windows,

small pipes and some of the dormers and chimneys. Without a properly sized and positioned polygon for those obstacles it is pretty much impossible to run the algorithm for many of the roofs.

Roof surface selection

Another thing to developed is a way to select the relevant roof surfaces that belong to a certain address and to find the most suitable roof surface. This aspect was already mentioned in the future steps of the algorithm, but it plays predominantly a very big role in the large scale implementation. The algorithm will have to pick the right roof surfaces out of many and that can be a challenging task. Once again can information about the actual roof services maybe bring the solution.

Roof polygon inaccuracy

An aspect related the previous things this is the accuracy of the roof polygons. Drawing them by hand had an inaccuracy of about half a meter. Doing that automatically will probably not make that much better. This means that a solution has to be found to deal with this inaccuracy. Perhaps data about the actual sizes of the roof shapes can be found and used.

10.2.4. Small scale implementation

A small scale implementation is defined as a single roof section. This can be an interesting addition to the software that Solar Monkey develops. The option to let the software suggest a possible PV system for a selected roof surface can be a unique service. There are however still a few steps that have to be taken for this to be realized.

Drawing the roof polygons

The big step to take is to make sure that the roof polygons are obtained. An algorithm that does that automatically is under development by the company Readaar. It is however also possible to have a semi-automatic design version, in which the roof polygons still have to be drawn. One of the major troubles with this is that the roof obstacles often are not visible on the AHN layer. They have to be estimated by looking at an aerial photograph, but those are often taken at an angle. If those photographs would be at an exact vertical, they can be used to draw the obstacles. The photograph can scaled to the same size as the AHN layer be placed on top of it. The obstacles can then be drawn easily and correctly. This procedure is illustrated in figure 10.2.

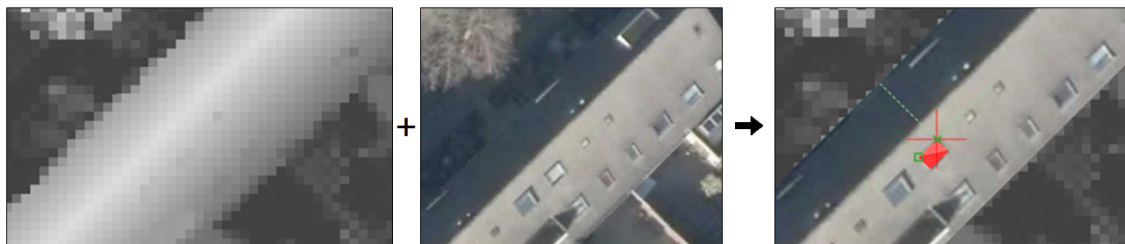


Figure 10.2: The AHN layer and a aerial photograph could be combined to draw the roof polygons.

The photograph used in figure 10.2 is not perfectly vertical, which makes this process not yet viable. If the used photo is completely vertical, it would possible to simply draw the roof obstacles at the right position and scale. The customer will be asked to indicate the obstacles on the roof. This can be done very fast if the vertical photo and the AHN layer are placed on top of each other. The customer then just draws the obstacles on top of the photo with easy to use drawing tools. Once the customer is satisfied it can save the obstacles and those will be exported as a shapefile and immediately used as input for the algorithm. The step to enable the drawings and to export to shapefiles like in QGIS will still take some programming, but the principle of obtaining the roof obstacles should work.

Speed optimizations

The current algorithm is still just a prototype. This means that several aspects require attention before it can be implemented. One of those aspects is the optimization of the calculation speed. This for the most part be achieved with more efficient coding. The algorithm does several operations that can be coded in a lot of different ways. Not too much attention was placed in trying to optimize this. The operations are documented and a more proficient coder can without a doubt find smarter ways to perform those operations with more efficient coding.

Parameter optimizations

Another thing yet to be optimized are some of the parameters chosen to run the algorithm. Those parameters are the panel sections to use, the step sizes and the buffer areas. Most of these will be determined how fast the algorithm can compute after speed optimizations. Another parameter is the layout grading. Perhaps the option can be enable to allow the customer to indicate how much is cared about aesthetics. The algorithm could provide more performance based layouts, which would score lower on the aesthetics scale.

If these steps are taken it is possible to have a first version of an automated PV system design algorithm. The customer can then enter the amount of panels that is desired and then ask the software to make a suggestion. The obstacles will have to be indicated, but that will only take up to a minute with proper drawing tools. It can be a very good addition to the software provided to installers.

Bibliography

- [1] S. Dale. *BP Statistical Review of World Energy June 2016*. BP, 2016.
- [2] J. Conti. *INTERNATIONAL ENERGY OUTLOOK 2016*. Energy Information Administration, 2016.
- [3] B. Kahn. *Antarctic CO2 Hit 400 PPM for First Time in 4 Million Years*. Climate Central, 2016.
- [4] J. Schmidt. *Paris Climate Agreement Explained: What's in it and where is it taking us?* Natural Resources Defense Council, 2015.
- [5] A. McCrone, U. Moslener, F. d'Estais, E. Usher, and C. Grüning. *Global Trends in Renewable Energy Investment 2016*. Frankfurt School – UNEP Collaborating Centre for Climate & Sustainable Energy Finance, 2016.
- [6] C. Philibert. *Technology Roadmap Solar Photovoltaic Energy*. International Energy Agency, 2014.
- [7] G. Masson and M. Brunisholz. *2015 SNAPSHOT OF GLOBAL PHOTOVOLTAIC MARKETS*. International Energy Agency, 2016.
- [8] A. Smets, K. Jäger, O. Isabella, R. van Swaaij, and M. Zeman. *Solar Energy The Physics and engineering of photovoltaic conversion technologies and systems*. UIT Cambridge, 2016.
- [9] P. Winterman. *Veel fouten bij installatie zonnepanelen*. Algemeen Dagblad, 2015.
- [10] G. Turk. *SolarSource: A general framework for evaluating rooftop solar potential*. Independent Work Report, 2015.
- [11] B. Al. *Top 15 Roof Types, Plus Their Pros & Cons 2017 – Read Before you Build!* Available at <https://www.roofcostestimator.com/top-15-roof-types-and-their-pros-cons/>, Accessed 2-September-2017.
- [12] S. Ekici and M. Kopru. *Investigation of PV System Cable Losses*. International Journal of renewable energy research, 2016.
- [13] E. Bellini. *Netherlands to extend net metering to 2023*. Available at <https://www.pv-magazine.com/2017/07/17/netherlands-to-extend-net-metering-to-2023/>, Accessed 2-September-2017.
- [14] A. Breukel, C. van Dijk, and K. Spee. *The Relative Importance of Aesthetics in the Adoption Process of Solar Panels in the Netherlands*. Conference: Engineering education for sustainable development at Bruges, 2016.
- [15] J. Lijzenga and J. Boertien. *De rol van woningcorporaties op de woningmarkt - een WoON 2015-verkenning*. Companen, 2016.
- [16] A. Veenstra. *Ruimte voor zonne-energie in Nederland 2020-2050. Analyse van ruimtelijke groeikansen en knelpunten voor zonne-energie toepassingen in Nederland*. Holland Solar, 2015.
- [17] G. Martins and R. Dell. *Solving the pallet loading problem*. Elsevier, 2007.
- [18] E.G. Birgin, R.D. Bogato, and R. Morabito. *An effective recursive partitioning approach for the packing of identical rectangles in a rectangle*. Journal of the operational research society, 2010.
- [19] E.G. Birgin. *Recursive partitioning approach for the Manufacturer's Pallet Loading Problem*. Available at <http://lagrange.ime.usp.br/~lobato/packing/>, Accessed 2-September-2017.
- [20] *Impact of shading*. Available at <http://www.greenrhinoenergy.com/solar/performance/shading.php>, Accessed 2-September-2017.

- [21] P. Holberg. *Solmetric powerpoint presentation*. Solmetric, 2011.
- [22] M. Kumar, M. Husian, N. Upreti, and D. Gupta. *Genetic algorithm: review and application*. International Journal of Information Technology and Knowledge Management July-December, 2010.
- [23] E. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley & Sons Ltd, 1997.
- [24] M. Stolk. *Huishoudens over zonnepanelen - Bezit, intentie tot aankoop, mogelijke drempels en informatiebehoefte, MilieuCentraal*. Milieucentraal, 2014.
- [25] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II*. Springer, 2000.
- [26] Unknown. *Pareto efficiency*. Available at https://upload.wikimedia.org/wikipedia/commons/b/b7/Front_pareto.svg, Accessed 2-September-2017.
- [27] F. Vignola, F. Mavromatakis, and J. Krumsick. *Performance of PV inverters*. Conference: American Solar Energy Society, 2008.
- [28] Energysage. *String inverters vs. Microinverters vs. Power optimizers*. Available at <https://www.energysage.com/solar/101/string-inverters-microinverters-power-optimizers/>, Accessed 2-September-2017.
- [29] J.A. Bondy and U.S.R. Murty. *Graph theory with applications*. Elsevier Science Publishing Co., Inc., 1982.
- [30] R.J. Wilson. *Introduction to graph theory, 4th edition*. Addison Wesley Longman Limited, 1996.
- [31] J.H.C. Barreto. *Optimizing inverter sizing for PV systems according to their installation characteristics*. Unpublished master thesis, 2015.
- [32] *latitudelongitude*. Available at http://www.geographyalltheway.com/ks3_geography/maps_atlases/imagesetc/latitudelongitude.png, Accessed 2-September-2017.
- [33] *Alles over gps*. Available at <http://www.allesovergps.nl/rechthoekig.jpg>, [Online; accessed 2-September-2017].
- [34] G.R. Harik and F.G. Lobo. *A parameter-less genetic algorithm*. GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, 1999.

Appendix A - Genetic Algorithm

In this section the genetic algorithm is explained. Attempts were made to develop a genetic algorithm that would serve as a finite panel placement algorithm. The concept of what the algorithm was doing is discussed here.

The five general steps that take place in the genetic algorithm are: initialization, evaluation, selection, crossover and mutation. First an initial population is created. This consists of possible solutions to the problem. These are then all evaluated and then a selection is performed based on this evaluation. New solutions are then created by manipulating the selected solutions and by merging their properties. This process increases the overall value of the solutions. This is continued until a stopping criterion is met. This can be that a solution reaches a certain level, after a certain amount of operations or after a certain period of time.

Initialization

The first phase of the genetic algorithm is the initialization in which the initial population is created. This population consist of pseudo random solutions. The size of the population is largely dependent on the problem and the problem search space. Several different population size were tried. Also trying different sizes depending on the roof. Eventually a technique was used from what is called a parameter-less genetic algorithm, which was first proposed by Harik and Lobo in their paper [34]. In this method not a fixed population size is used, but a size that increases with the complexity of the problem. It starts with a small population and increases this as long as a satisfying solution is not found. This solved the problem of not being able to pick a right population size for all scenario's.

In order to generate a random solution for the problem, the first step is to determine how a solution is represented. In this case a solution is one or more sections of panels in a certain orientation, in a certain configuration and at a certain position on the roof. This is represented as a list with first the panel section configuration, then the orientation and then the position. An example is $[[2, 3], 1, [40, 100]]$, which is a 2x3 section in landscape orientation with the center point positioned at coordinate (40, 100). The orientation for portrait is 2. For a solution with more panel section another section with its position and orientation is just added to the list.

When a solution is generated a random configuration, orientation and position is chosen. The solution consist of one or two panel sections that both add up to the required amount of panels. All within the boundaries of the roof surface and the design preference.

Evaluation

A very important step in the genetic algorithm is the evaluation step. This is done by using a fitness function, which will evaluate the solutions and assign a rating to them. This fitness function is the same as described in the layout grading in chapter 6.

Selection

After each solution is evaluated, a selection of individuals is made. This selection is based on the fitness of the solutions. The selected solutions will be subjected to crossover and mutation operators that will push the population towards an optimum.

There are many different selection methods that can perform this operation. The goal is to select in such a way that the average quality of the population increases, while simultaneously maintaining a diverse population. If the population is not kept diverse enough, the population will converge very quickly to an optimum that is likely to be a local optimum instead of the global optimum that his desired.

When using multiple objectives, a method for selection is a rank selection. In this method the solutions are sorted based on their rank, which are determined by an elitist non-dominated sorting method [25]. A top of the solutions is then selected to form new solutions.

Crossover

One of the operations that takes place is a crossover between two solutions. With this crossover two new solutions are created, which are called the children. The two solutions that were used are called the parents. The principle is that the parents will swap out genes in order to make new solutions. The two parents are randomly selected. In this genetic algorithm the crossover consists of a swapping of positions. This concept is illustrated in figure 3.

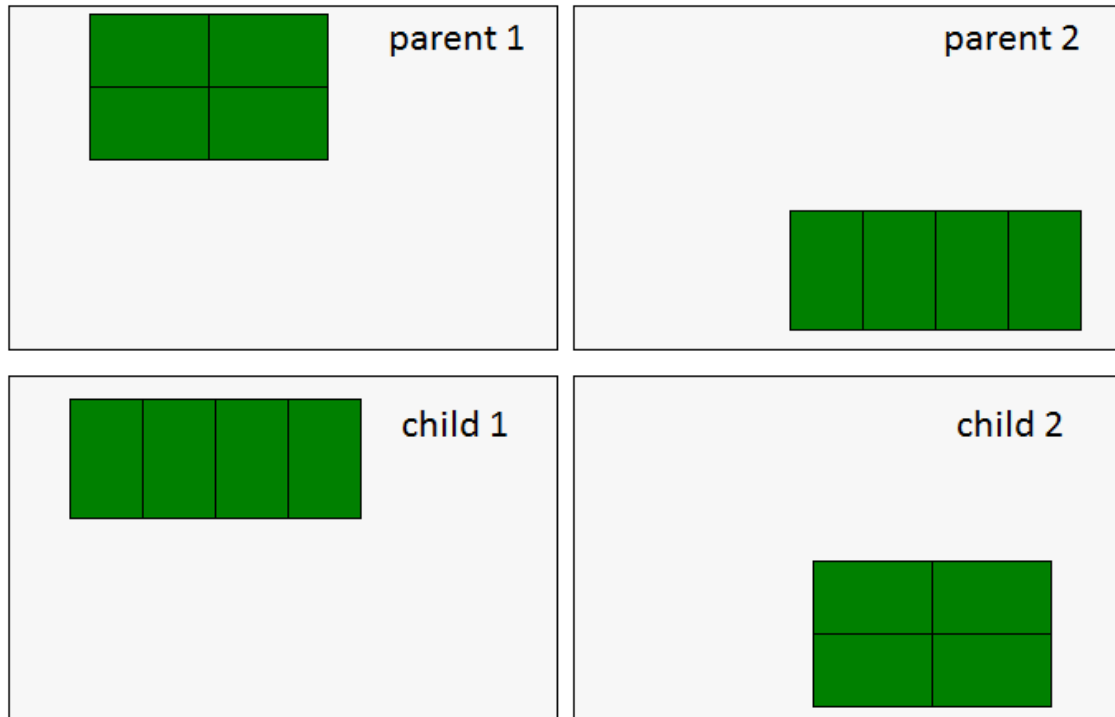


Figure 3: With the crossover the parent solutions swap position, which creates two new solutions.

Mutation

Mutation in this genetic algorithm is done by performing a local search. This means that the panel section is allowed to take a step in four possible directions: up, down, left or right. The algorithm calculates the value of the panel section in all of those four positions and the step that will result in the best performance will be taken. This process continues until no step can be taken that will improve the position of the panel section. This process is a local optimization, as the panel section will be placed in a local optimum position. A local optimum in this case means the area near the initial position that has the least amount of shading. The algorithm does the local search twice, once for each orientation of the panels. This means that it is also optimized for landscape and portrait orientation. The step size of this local search can be varied.

Genetic algorithm results and evaluation

The genetic algorithm was tested with various parameter settings. These parameter settings consist of the selection method, the initialization, the step size of the local search and adjustment of the fitness function. The results seemed very promising at first. Solutions were made that consisted of one or two sections that were placed reasonably well on the roof.

A concern was the fact that this method was difficult to expand to solutions with more than two panel sections. Another big reason is that the algorithm seems to be inefficient in what it tries to accomplish. In order to understand that it is important to think about what makes a good solution and how the genetic algorithm moves towards solutions that fit that criterion.

A good solution consist of panel sections that together make up the required amount of panels and are placed on good areas on the roof. By good areas an area with a low amount of shading is meant. The genetic algorithm finds these solutions with the crossover and mutation functions. With the mutation a local optimum is found for a certain panel section. This is however not necessarily the best panel section for that particular local optimum. Sometimes a large panel section ends up in a local optimum that only consists of small not shaded area. It also happens that a small panel section ends up in large not shaded area. The crossover function swaps positions of panels that are locally optimized and this makes sure that at some point a panel section ends up in the local optimum where it fits best. This concept is illustrated in figure 4.

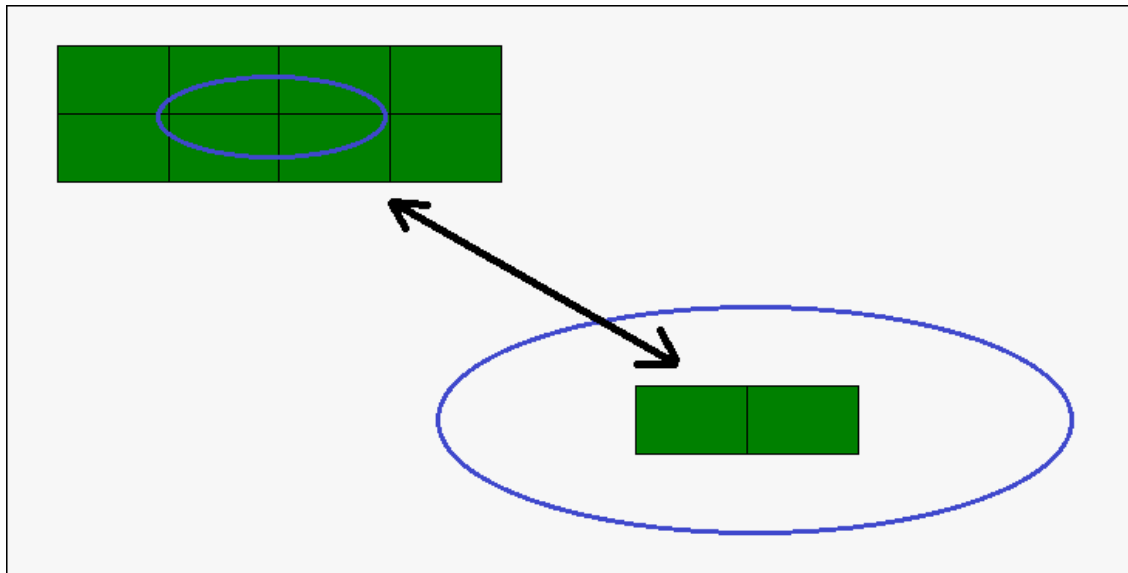


Figure 4: Crossover places panel sections in the local optimum where it fits best. The local optimum is indicated in blue.

This image illustrates the concept of matching the panel sections with a local optima in which it fits best. The final solution than consists of panel sections that are placed at the right position in this manner. A new method was developed to reach this goal that was found to be more efficient.

Appendix B - Pseudo code of Algorithms

Algorithm 1 Determine Panel Orientation

```
1: calculate distances between subsequent roof polygon points
2: get polygon points longest edge
3: panel orientation  $\leftarrow \arctan(\Delta x/\Delta y)$ 
4: return panel orientation
```

Algorithm 2 Maximum panel placement Algorithm

```
1: if pitch angle == 0 then
2:   if east west == False then
3:     preserve row distance
4:     panel orientation  $\leftarrow$  Algorithm 1 Determine Panel Orientation
5:   if east west == True then
6:     panel orientation  $\leftarrow$  east west
7: panel grid  $\leftarrow$  create panel grid that covers roof polygon
8: for i in range(0, panel length, step size) do
9:   for j in range(0, panel width, step size) do
10:    Shift panel grid
11:    Discard panels not within roof polygon minus buffer roof edge
12:    Discard panels that intersect with obstacles plus buffer obstacles
13:    Save configuration and number of panels
14: solution  $\leftarrow$  configuration with most panels
15: return solution
```

Algorithm 3 Get Roof Positions

```

1: if Trivial then
2:   step size  $\leftarrow$  25
3:   buffer roof edge  $\leftarrow$  half smallest panel section edge
4:   roof positions  $\leftarrow$  step size spaced point grid that covers roof polygon
5:   for each positions do
6:     discard if point not within roof polygon minus roof edge buffer
7:     discard if point intersects with obstacle plus obstacle buffer
8: if Large then
9:   step size  $\leftarrow$  10
10:  find roof polygon centroid
11:  roof positions  $\leftarrow$  step size spaced grid of 15 by 15 around centroid
12: if Combined then
13:  step size  $\leftarrow$  25
14:  buffer roof edge  $\leftarrow$  half panel width
15:  roof positions  $\leftarrow$  step size spaced point grid that covers roof polygon
16:  for each positions do
17:    discard if point not within roof polygon minus roof edge buffer
18:    discard if point intersects with obstacle plus obstacle buffer
19: return roof positions

```

Algorithm 4 Get Panel Sections

```

1: bounding box  $\leftarrow$  create bounding box of polygon
2: if pitch angle == 0 then
3:   only horizontal rows
4: if trivial then
5:   panel sections  $\leftarrow$  all sections that fit the panel range
6:   for each section do
7:     discard sections that don't fit bounding box
8: if Large then
9:   panel sections  $\leftarrow$  largest sections that fit bounding box
10: if Combined then
11:  panel sections  $\leftarrow$  all sections that can combine to make panel range
12:  for each section do
13:    discard sections that don't fit bounding box
14:  if there are obstacles then
15:    for each section (a, b) do
16:      also add [a, b+1] and [a+1, b] to panel sections
17: return panel sections

```

Algorithm 5 Finite panel placement Algorithm

```

1: roof polygon, roof obstacles, performance grid ← Algorithm 6 Shapefile To Adjusted Polygons And Performance Grid
2:
3: if pitch angle == 0 then
4:   preserve row distance
5:   panel orientation ← Algorithm 1 Determine Panel Orientation
6:
7: trivial roof points ← Algorithm 3 Get Roof Positions(trivial)
8: panel sections ← Algorithm 4 Get Panel Sections(trivial)
9: for each panel section do
10:   evaluate at each roof point
11: trivial solutions ← evaluated trivial sections
12:
13: large roof points ← Algorithm 3 Get Roof Positions(large)
14: panel sections ← Algorithm 4 Get Panel Sections(large)
15: for each panel section do
16:   evaluate at each roof point
17: large solutions ← evaluated large sections
18:
19: combined roof points ← Algorithm 3 Get Roof Positions(combined)
20: panel sections ← Algorithm 4 Get Panel Sections(combined)
21: for each panel section do
22:   evaluate at each roof point
23: combined solutions ← combination of all panel sections
24:
25: all solutions ← trivial solutions + large solutions + combined solutions
26: ranked solutions ← filtered and ranked all solutions
27:
28: return ranked solutions

```

Algorithm 6 Shapefile To Adjusted Polygons And Performance Grid

```

1: polygons WGS84 ← fiona(shapefile)
2: polygons RD ← pyproj.transform(polygons WGS84)
3: performance grid polygons ← create 50x50cm grid that covers roof polygon
4: grid point WGS84 ← pyproj.transform(grid points RD)
5: performance values ← calculate panel performance at each grid point WGS84
6: polygons ← polygons in RD + performance grid polygons
7: if pitch angle != 0 then
8:   rotate polygons with roof azimuth
9:   scale polygons with pitch angle correction
10: roof polygon ← polygons[0]
11: roof obstacles ← polygons[1: number of obstacles + 1]
12: performance grid polygons ← polygons[number of obstacles + 1 :]
13: return roof polygon, roof obstacles, performance grid polygons, performance values

```

Algorithm 7 Panel miss match v1

```

1: relative intensity  $\leftarrow$  get relative intensity ☞ should be the same for each panel
2: for each panel do
3:   get obstacles
4:  $P_A \leftarrow$  numpy.sum(relative intensity * obstacles A)
5:  $P_B \leftarrow$  numpy.sum(relative intensity * obstacles B)
6:  $P_{AB} \leftarrow$  numpy.sum(relative intensity * obstacles A * obstacles B)
7: panel miss match  $\leftarrow 1 - 2 \cdot P_{AB} / (P_A + P_B)$ 
8: return panel miss match

```

Algorithm 8 Panel miss match v2

```

1: get monthly sun tracks
2: get irradiance
3: for each panel do
4:   get obstacles
5:   get relative intensity (RI)
6: for each month do
7:   for each panel do
8:     pixels  $\leftarrow$  RI * sun track
9:     direct  $\leftarrow$  pixels * obstacles * (direct irradiance / numpy.sum(pixels))
10:    diffuse  $\leftarrow$  RI * obstacles * (diffuse irradiance / numpy.sum(RI))
11:    total  $\leftarrow$  direct + diffuse
12:    total per orientation  $\leftarrow$  sum the values at each orientation
13:  value A month  $\leftarrow$  sum irradiances of each panel per orientations
14:  value B month  $\leftarrow$  worst irradiance per orientation x number of panels
15: total A  $\leftarrow$  sum all monthly A values
16: total B  $\leftarrow$  sum all monthly B values
17: panel miss match  $\leftarrow 1 - totalA/totalB$ 
18: return panel miss match

```

Appendix C - Inverter sizing table

		Module Azimuth [°]																
		0	N	NNE	NE	ENE	E	ESE	SE	SSE	S	SSW	SW	WSW	W	WNW	NW	NNW
Tilt Angle [°]	0	111	119	124	156	204	270	322	377	90								
	15	111	117	124	152	192	232	270	322	377	0	N						
30	111	115	121	139	152	168	181	204	204	22.5	NNE							
45	111	115	117	124	126	131	142	159	159	45	NE							
60	111	108	109	109	109	115	126	145	145	67.5	ENE							
75	111	104	104	101	103	108	119	139	139	90	E							
90	111	103	101	98	98	104	117	136	136	112.5	ESE							
	111	101	100	97	98	103	115	134	134	135	SE							
	111	100	97	97	97	104	113	129	129	157.5	SSE							
	111	101	98	97	97	103	115	131	131	180	S							
	111	103	101	97	100	104	117	136	136	202.5	SSW							
	111	106	104	101	103	108	119	139	139	225	SW							
	111	109	109	108	111	115	124	142	142	247.5	WSW							
	111	111	117	119	126	131	142	159	159	270	W							
	111	117	119	136	152	163	181	197	197	292.5	WNW							
	111	117	124	152	186	224	259	307	307	315	NW							
	111	117	124	152	186	224	259	307	307	337.5	NNW							

Figure 5: Optimum inverter sizing % for different orientations and tilt angles. Table taken from [31]