RWTH AACHEN UNIVERSITY

Optimierung eines Heliostatenfeldes unter Einsatz eines genetischen Algorithmus

Diplomarbeit in Informatik
RWTH Aachen

von Georg Jennessen
März 2013

angefertigt bei

Prof. Dr. rer. nat. Erika Ábrahám
Theorie der hybriden Systeme (i2)

Prof. Dr. rer. nat. Martin Frank
Center for Computational Engineering Science

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Diese Arbeit hat in dieser oder einer ähnlichen Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

Aachen, im März 2013

Georg Jennessen

Inhaltsverzeichnis

1 Einleitung

Das Thema Energieversorgung wird weltweit kontrovers diskutiert. Die Einigung auf ein gemeinsames Konzept, speziell für eine umweltschonende Energieversorgung ist noch nicht in Sicht. Bei der 18. Weltklimakonferenz in Doha 2012 konnten sich die Teilnehmerstaaten lediglich auf eine Verlängerung des –ursprünglich von 1997-2012 gültigen- Kyoto-Protokolls bis 2020 verständigen. Immerhin bleibt damit der einzige, rechtlich bindende Vertrag, mit dem zumindest einige Industriestaaten zur Einhaltung vereinbarter Emissionsgrenzen verpflichtet werden können, bestehen. Außerdem besteht damit die Chance, bis 2015 ein neues Abkommen auszuhandeln, in dem sich auch die Entwicklungsländer verpflichten, ihre Treibhausgasemissionen zu reduzieren [1].

Die erneuerbaren Energien wurden im Jahr 2011 dennoch weiter ausgebaut. „Auch die Zahl der Länder mit Zielvorgaben für den Ausbau der erneuerbaren Energien hat sich erneut erhöht, ... Neben China, den USA und Deutschland fand das Wachstum der erneuerbaren Energien hauptsächlich in Spanien, Italien, Indien und Japan statt.“ [2]

Die 27 Mitgliedsstaaten der EU haben sich EU-intern bis 2020 verpflichtet, ihre Treibhausgasemissionen im Vergleich zum Jahr 1990 um 20 % zu senken, insbesondere durch Maßnahmen zu mehr Energieeffizienz,verstärkten Einsatz erneuerbarer Energien und Fortführung des EU-weiten Emissionshandels. [3]

Deutschland hat sich dabei zum Ziel gesetzt, „eine der energieeffizientesten und umweltschonendsten Volkswirtschaften der Welt zu werden.“ Um dies zu erreichen hat die Bundesregierung als Ergänzung zum Energiekonzept von Sep. 2010 im 6. Energieforschungsprogramm auf vielen Feldern neue Akzente gesetzt. „Von Bedeutung sind dabei vor allem: die klare Priorität bei der Förderung von Forschung und Entwicklung in den Bereichen Energieeffizienz und Erneuerbare Energien.“ Dabei wird die Notwendigkeit enger Zusammenarbeit nicht nur auf europäischer, sondern auch auf internationaler Ebene betont. Die langfristige Zielvorgabe bis 2050 für die erneuerbaren Energien lautet: „Ausbau auf einen Anteil von 60 % am Bruttoendenergieverbrauch (2020:18 %) bzw. 80 % am Bruttostromverbrauch (2020: mind. 35 %).“[4]
Neben dem Einsatz von Wind- und Biomassekraftwerken gewinnen dabei Solarkraftwerke zunehmend an Bedeutung.
Bei den Solarkraftwerken lassen sich zwei grundsätzliche Typen unterscheiden:

- Photovoltaikanlagen, die mittels Solarzellen Energie erzeugen, indem ein Teil der Sonnenstrahlung durch Ausnutzung des photovoltaischen Effekts in elektrischen Strom umgewandelt wird, und
- Thermische Solarkraftwerke, die zur Stromerzeugung über einen Absorber die Wärme der Sonnenstrahlen nutzen.

Thermische Solarkraftwerke erreichen zwar in der Regel höhere Wirkungsgrade als Photovoltaikanlagen, verursachen dafür aber höhere Betriebs- und Wartungskosten und sind

dadurch im Verhältnis zur gewonnenen Energie noch sehr teuer [5]. Sie sind nur in besonders sonnenreichen Regionen der Erde wirtschaftlich einsetzbar, was eine gute Zusammenarbeit mit einem Partnerland zur Voraussetzung hat.
Allerdings sind sie deutlich umweltschonender als Kraftwerke, die die Energie durch Verbrennung produzieren. Außerdem liefern sie Strom nach Bedarf und können lt. einer Studie des EASAC (European Academies Science Advisory Council) das Stromnetz zuverlässig stabilisieren.
Ein weiterer Vorteil dieser Art der Energiegewinnung besteht darin, dass kaum Landnutzungskonkurrenz entsteht, weil diese Kraftwerke v.a. in Trockenzonen der Erde genutzt werden.
Und schließlich bieten sie Ländern wie Deutschland den Vorteil, dass die bereits gemachten Erfahrungen beim Einsatz von anderen Kraftwerken auch hier genutzt werden können. Thermische Solarkraftwerke bündeln die Sonnenstrahlen entweder auf ein Absorberrohr das in der Brennlinie einer Parabolrinne liegt, siehe Abbildung 1(a) oder auf einen zentralen Absorber. Bei Parabolspiegeln, siehe Abbildung 1(b) liegt dieser Absorber direkt vor dem Spiegel in seinem Brennpunkt. Ein weiteres Solarkraftwerk mit einem zentralen Absorber ist der Solarturm, welcher Forschungsobjekt der vorliegenden Arbeit ist.

(a) Paralbolrinne

(b) Parabolspiegel

Abbildung 1: Thermische Solarkraftwerke

Quelle: DLR, Novatec Solar

1.1 Solarturm

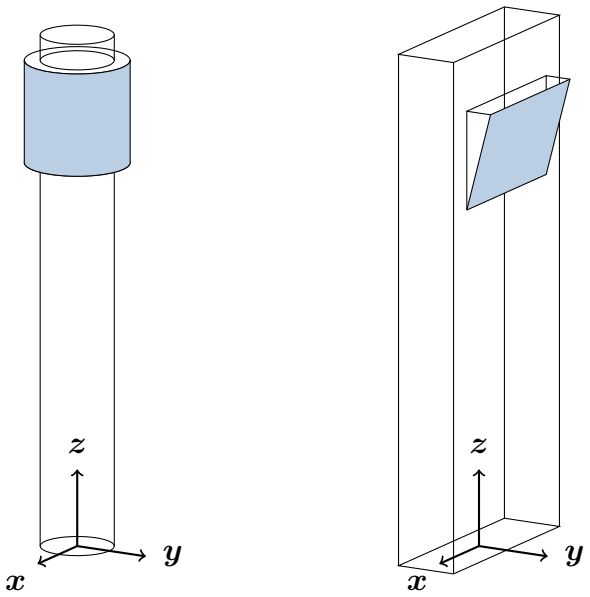
Hunderte bis tausende Spiegel sind vor einem Turm aufgestellt und richten sich je nach Sonneneinstrahlung so aus, dass die reflektierten Sonnenstrahlen einen Receiver treffen, der an der Spitze des Solarturms angebracht ist. Die sich automatisch ausrichtenden Spiegel werden Heliostate genannt [6]. Ein Solarturm arbeitet ähnlich wie Gas- oder Kohlekraftwerke. Hierbei wird jedoch die Brennkammer durch den Receiver ersetzt, der ein Wärmeträgermedium erhitzt. Dieses ist entweder flüssiges Nitratsalz, Wasserdampf oder Heißluft. Die Jahrzehnte lange Erfahrung mit den herkömmlichen Kraftwerken und die Anforderung an effektiver Gewinnung von erneuerbaren Energien kann so optimal kombiniert werden.

Tabelle 1: Standorte von Solartürmen

Name	Ort	Receiver-Typ	Leistung
Planta Solar 10 (PS10)	Seville, Spanien (37.442°,-6.250°)	einseitig	11 MW
Planta Solar 20 (PS20)	Seville, Spanien (37.440°,-6.260°)	einseitig	20 MW
GEMASOLAR	Seville(Spanien) (37.560°,-5.331°)	360°	19,9 MW
Jülich Solar Tower	Jülich (Deutschland) (50.913°,6.387°)	einseitig	1,5 MW

Standorte Einige dieser Solarturmkraftwerke sind in Amerika oder Spanien im Einsatz. Auch in Deutschland (Jülich) wurde ein Kraftwerk zu Versuchs- und Demonstrationszwecken gebaut. Einige der Standorte sind in der Tabelle 1 mit Koordinaten aufgelistet.

Es gibt zwei Typen von Receiver, siehe Abbildung 2



(a) 360 ° Receiver

(b) Einseitiger Receiver

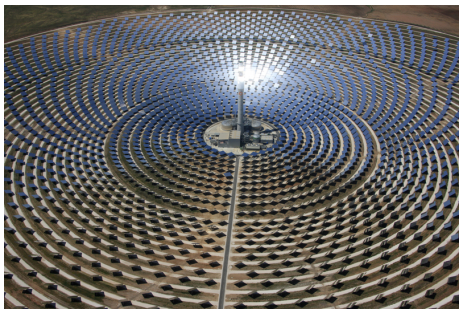
Abbildung 2: Vergleich von Receivertypen

Einseitiger Receiver In Regionen, die vom Äquator weit entfernt sind, kommt die Sonneneinstrahlung größtenteils aus einer Himmelsrichtung. In Deutschland steht die Sonne z.B. tagsüber im Süden. Die Heliostate sind dementsprechend in diese Richtung

ausgerichtet, der Receiver nach Norden. Da der Receiver nur von genau einer Seite bestrahlt wird, hat er die Form eines flachen Rechtecks.

360 ° Receiver Im Gegensatz zu den einseitigen Receivern können die zylindrischen Receiver von allen Seiten bestrahlt werden. So können die Sonnenstrahlen, unabhängig vom Sonnenstand zu jeder Tageszeit auf den Receiver gelenkt werden.

(a) Planta Solar 10



(b) GEMASOLAR

Abbildung 3: (a) Planta Solar 10 (PS10) in Seville, Spanien mit einseitigem Receiver (b) GEMASOLAR in Seville, Spanien mit 360° Receiver
Quelle: flickr, DLR

1.2 Ziel dieser Arbeit

Das Ziel dieser Arbeit ist, einen Optimierungsalgorithmus zu entwickeln, der eine optimale Position der Heliostate berechnet, mit dem Ziel, dass die am Receiver ankommende Strahlung im Jahr maximiert wird.
Als Optimierungs-Algorithmus wird dazu der Genetische Algorithmus [7] verwendet. Das Modell musste dazu entsprechend angepasst werden.

1.3 Gliederung

Die vorliegende Arbeit ist folgendermaßen gegliedert. In Kapitel 2 wird das zu Grunde liegende Modell zur Berechnung der Leistung eines Solarturms beschrieben. In Kapitel 3 wird auf das Prinzip des verwendeten Genetischen Algorithmus eingegangen und wie dieser auf das Problem der Optimierung von Solartürmen übertragen wurde. Um die Funktionsweise des Algorithmus zu testen, wurden verschiedene einfache Zielfunktionen verwendet. In Kapitel 4 wird die Struktur und ausgewählte Klassen der Software beschrieben. Die Ermittlung der optimalen Parameter für den Genetischen Algorithmus und die verwendete Parallelisierung, so wie die Anwendung der Software auf einen konkreten Solarturm wird in Kapitel 5 beschrieben. Im abschließenden Kapitel 6 werden die Ergebnisse zusammengetragen und Ausblicke bezüglich weiterer Erweiterungen und Anpassungen der Optimierung gegeben.

2 Modell

Ein Solarfeld besteht aus einer Menge von Heliostaten, welche die Sonnenstrahlen auf einen am Turm angebrachten Receiver lenken. Anhand von Koordinaten der Heliostate und des Solarturms und seiner Parameter wie Höhe, Größe und Neigung des Receivers kann mit Hilfe eines Raytracing Modells berechnet werden, wie viel Strahlungsenergie der Receiver einfängt. Das Modell [8] berechnet anhand der gegebenen Parameter die Jahresleistung des Solarturmkraftwerkes. Im folgenden wird das verwendete Modell vorgestellt, welches durch das Monte Carlo Raytracing Tool SolTRACE [9] verifiziert werden kann.

2.1 Raytracing Modell

2.1.1 Sonnenposition

Zunächst muss die Position der Sonne berechnet werden, um den Weg zu bestimmen, den ein Sonnenstrahl von der Sonne über den Spiegel zum Receiver nimmt. Das Raytracing Modell benutzt dazu einen Algorithmus [10], der zudem noch die Sonnenauf- und Sonnenuntergangszeiten liefert.

2.1.2 Ausrichtung der Heliostate

Ein Heliostat verfolgt den Sonnenstand, um die Sonnenstrahlen immer an einen fixen Punkt, den am Turm befestigten Receiver zu lenken. Das Modell berechnet dazu anhand der Sonnenpositionen die genaue Ausrichtung des Spiegels.

2.1.3 Direktnormalstrahlung DNI

Die Direktnormalstrahlung DNI ist die Energie der Sonnenstrahlen in kWh/m^2, die zu einem bestimmten Zeitpunkt und Ort auf einen Spiegel trifft. Diese Werte werden dann für die weitere Berechnung verwendet, um die Energie zu bestimmen, die am Receiver ankommt. Für die Ermittlung der DNI Werte wird das Meteorological Radiation Model (MRM) [11] verwendet.

2.2 Verluste

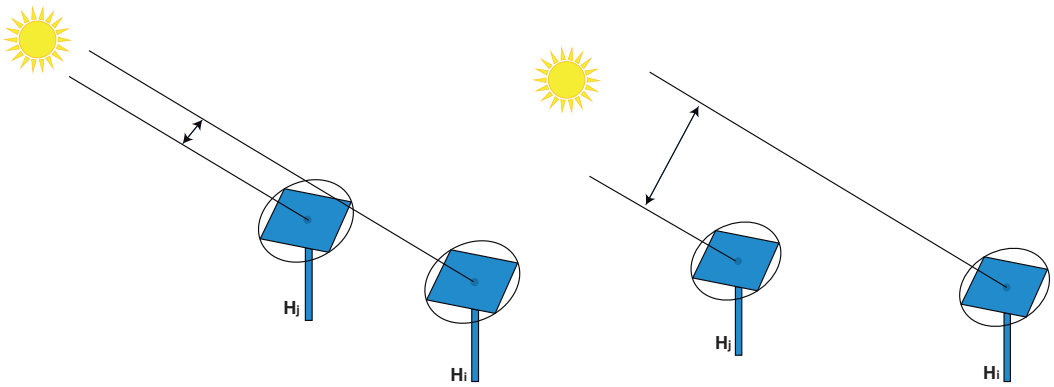
Es gibt verschiedene Gründe, weshalb ein Sonnenstrahl von der Sonne über einen Spiegel nicht oder nur teilweise zu dem Receiver gelangt.

2.2.1 Kosinus-Effekt

Die Heliostate werden automatisch so ausgerichtet, dass die ankommenden Sonnenstrahlen in Richtung des Receivers reflektiert werden. Jedoch wird die projizierte Fläche durch die Neigung des Heliostats verringert, sodass ein Kosinus-Effekt entsteht [12]. Dieser Effekt ist abhängig von der Sonnenposition und der Position der Heliostaten.

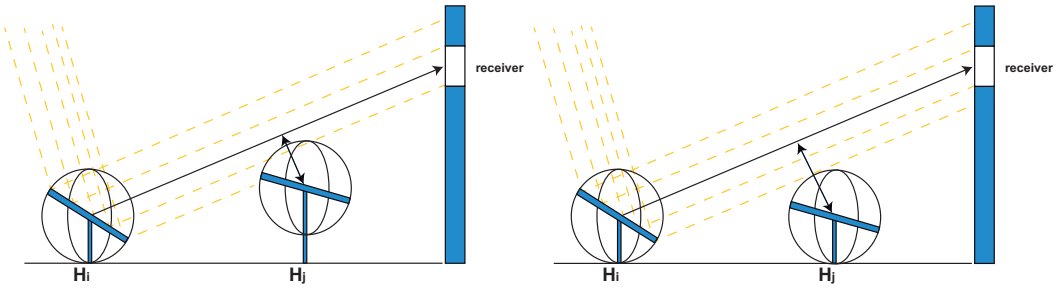
2.2.2 Blockierung und Verschattung

Die Sonnenstrahlen treffen auf die Oberfläche der Spiegel und werden in Richtung des Receivers reflektiert. Jedoch kann der Weg von der Sonne zu einem Heliostat durch Objekte wie dem Turm oder andere Heliostate blockiert sein. Sie werfen also einen Schatten auf den Spiegel, siehe Abbildung 4. Auch der Weg von einem Spiegel zum Receiver kann durch andere Heliostate blockiert sein, siehe Abbildung 5. Je nach Position der Heliostate entstehen so im gesamten Verlauf des Tages mehr oder weniger Verluste.



(a) H_j beschattet H_i (b) H_j beschattet H_i nicht

Abbildung 4: Beispiel für die Verschattung eines Heliostaten



(a) H_j blockiert H_i (b) H_j blockiert H_i nicht

Abbildung 5: Beispiel für die Blockierung der reflektierten Strahlung eines Heliostaten

2.2.3 Heliostat Reflexion

Die Oberfläche des Spiegels reflektiert den größten Teil der Sonnenstrahlen, ein kleiner Teil wird absorbiert. Gemäß [13] werden etwa 87% der Sonnenstrahlen reflektiert und 13% von der Oberfläche absorbiert.

2.2.4 Atmosphärische Abschwächung

Die Sonnenstrahlen werden durch die Atmosphäre der Erde abgeschwächt. Durch Staubpartikel in der Luft, besonders in Bodennähe wird die Leistung noch weiter reduziert. Je weiter ein Heliostat vom Receiver entfernt ist, desto höher ist der Verlust [14], der exponentiell mit der Distanz wächst.

2.3 Erweiterung des Modells

2.3.1 Zielführungsfehler

Die Sonnenstrahlen kommen nicht nur aus dem Zentrum der Sonne, sondern mit einer gewissen Wahrscheinlichkeitsverteilung treffen auch Sonnenstrahlen vom Rand der Sonne auf die Spiegel. Diese Strahlen werden mit einem anderen Winkel an der Oberfläche der Spiegel reflektiert. Dieser Störungswinkel wird zusätzlich noch durch Unregelmäßigkeiten auf der Spiegeloberfläche und Ausrichtungsfehler verstärkt, so dass ein Austrittsstörungswinkel σ vorliegt. Insgesamt kann man sich also die Refexion am Spiegel durch einen Kegel vorstellen, auf dessen Grundfläche eine Gauß'sche Wahrscheinlichkeitsverteilung die umgelenkte Richtung des Strahls angibt [15], siehe Abbildung 6.

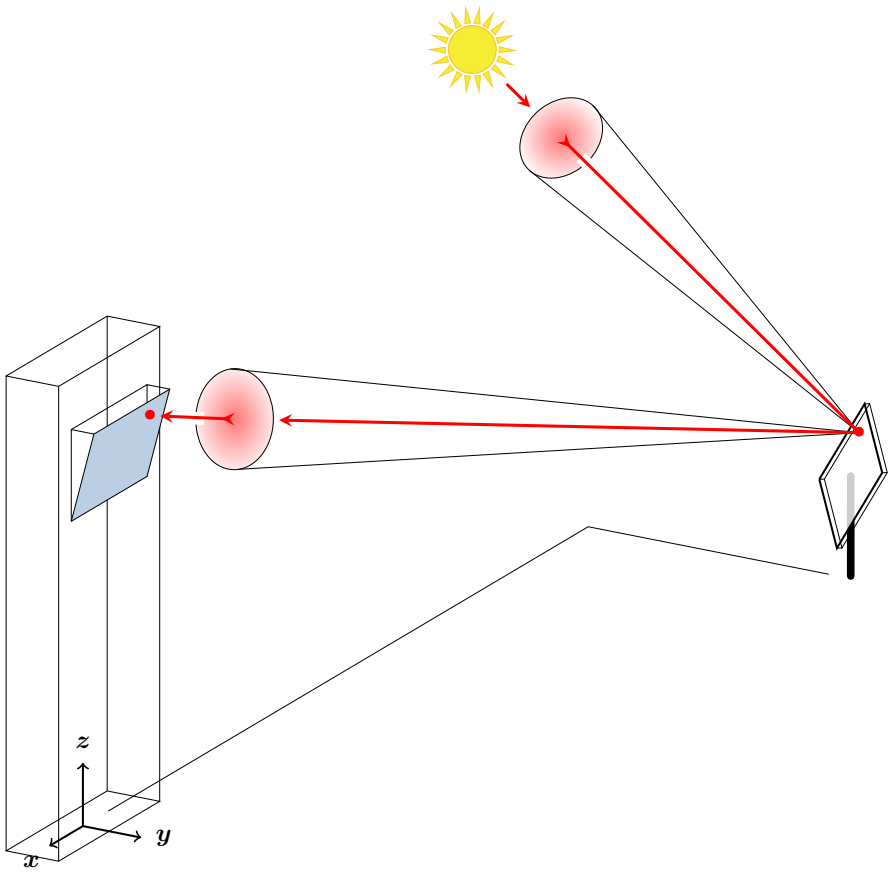


Abbildung 6: Zielführungsfehler, Berücksichtigung nicht paralleler Sonnenstrahlen

Das Modell überprüft, welcher Anteil eines jeden Strahls im Receiver ankommt. Dieser entspricht gerade dem Integral über die ankommende Normalverteilung des Strahls in dem Receiver, siehe Abbildung 7. Das Flächenintegral lässt sich ausdrücken durch

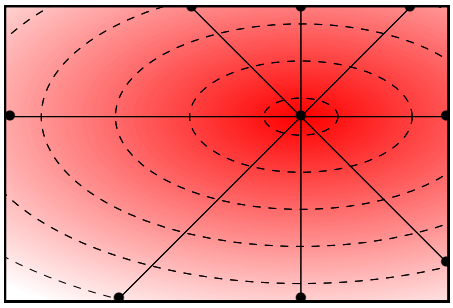
$$\eta_{\text{ziel}} = \frac{1}{2\pi} \int_0^{2\pi} \int_0^{\partial \text{rec}(\varphi)} \frac{1}{\sigma\sqrt{2\pi}} \, \mathrm{e}^{-\frac{1}{2}\left(\frac{\theta}{\sigma}\right)^2} \, \mathrm{d}\theta \, \mathrm{d}\varphi. \tag{1}$$

Dabei bezeichnet θ den Winkelabstand zum optimalen Strahl, welcher beschränkt ist durch den von φ abhängigen maximalen Winkelabstand $\partial\text{rec}(\varphi)$ bis zum Receiverrand. Das innere Integral kann mittels der Gaußschen Fehlerfunktion exakt bestimmt werden,

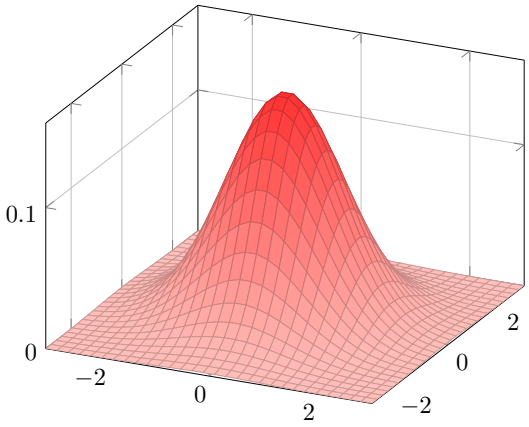
$$\eta_{\text{ziel}} = \frac{1}{2\pi} \int_0^{2\pi} \operatorname{erf}\left(\frac{\partial \text{rec}(\varphi)}{\sqrt{2}\sigma}\right) \, \mathrm{d}\varphi. \tag{2}$$

Das verbleibende äußere Integral wird numerisch mittels der zusammengesetzten Trapezregel gelöst. Dazu wird das Integrationsintervall von 0 bis 2π in n gleich große Teilintervalle unterteilt, so dass sich ergibt

$$\eta_{\text{ziel}} \approx \frac{1}{2\pi} \sum_{j=1}^{n} \operatorname{erf}\left(\frac{\partial \text{rec}(j \cdot \frac{2\pi}{n})}{\sqrt{2}\sigma}\right). \tag{3}$$



(a) Gauß'sche Wahrscheinlichkeitsverteilung auf dem Receiver



(b) Gauß'sche Wahrscheinlichkeitsverteilung

Abbildung 7: Zielführungsfehler

2.3.2 Indexstruktur - Gitter

Das Raytraycing Modell wurde um eine zweidimensionale Indexstruktur, dem Grid erweitert, um eine bessere Performanz zu erhalten. Das Prinzip dieser Indexstruktur ist ein Punktraster, welches über das Heliostaten-Feld gelegt wird, siehe Abbildung 8. Jeder

Heliostat kann so über die Koordinaten der Rasterpunkte ermittelt werden und untereinander räumliche Beziehungen erfasst werden. Zum Beispiel lässt sich schnell ermitteln, welche Heliostate zwischen dem Solarturm und einem beliebigen anderen Heliostat liegen um so zu prüfen, ob ein Sonnenstrahl auf dem Weg vom Spiegel zum Turm blockiert wird. Da der Genetische Algorithmus diese Indexstruktur auch verwendet, wird in 4.2 auf die technische Umsetzung genauer eingegangen.

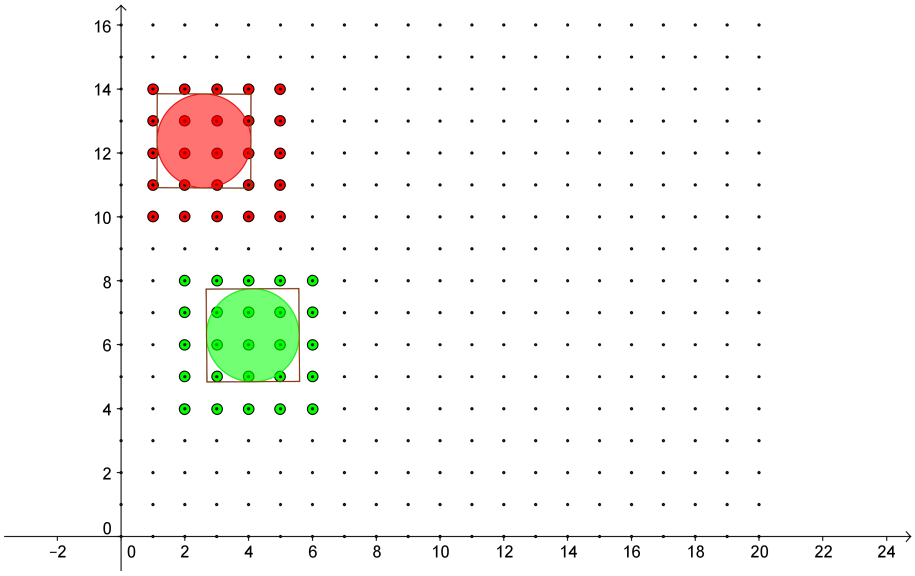


Abbildung 8: Speicherung der Objekte in einem solGrid

2.4 Berechnung der Energie

Gegeben ist ein Feld mit N Heliostaten H_i, mit jeweils einer Länge ℓ_i und Breite w_i. Die Gesamtfläche aller Spiegel ist

$$A_{\text{helios}} = \sum_{i=1}^{N} A_i = \sum_{i=1}^{N} \ell_i \cdot w_i. \tag{4}$$

Für eine zeitabhängige Direktnormalstrahlung I_{DNI} und den beschriebenen Verlusten, dem Kosinus-Effekt η_{cos}, der Blockierung und Verschattung η_{sb}, der Heliostaten Reflexion η_{ref}, der atmosphärische Abschwächung η_{aa} und dem Zielführungsfehler η_{inc} kann nun die Jahresenergie E_{year} berechnet werden:

$$E_{\text{year}} = A_{\text{helios}} \cdot \sum_{d=1}^{365} \left(\int_{\text{sunrise}}^{\text{sunset}} I_{\text{DNI}}(t) \cdot \sum_{i=1}^{N} \eta_{\text{cos},i}(t) \cdot \eta_{\text{sb},i}(t) \cdot \eta_{\text{ref},i}(t) \cdot \eta_{\text{aa},i}(t) \cdot \eta_{\text{inc},i}(t) \, \mathrm{d}t \right). \tag{5}$$

Das Modell benutzt zum Lösen dieses Integrals die Gauß-Quadratur.

3 Optimierung

Die Positionen der Heliostate sind dafür ausschlaggebend, wie viele Sonnenstrahlen den Receiver erreichen. Verschiedenste in Kapitel 2 vorgestellte Effekte reduzieren die Effizienz des Solarturms. Durch „intelligentes Ausprobieren“ soll nun eine möglichst optimale Positionierung der Heliostate berechnet werden, sodass sich möglichst wenige Heliostate gegenseitig beeinflussen und auf ein Kalenderjahr gerechnet die beste Ausnutzung erreicht wird. Zur Berechnung der Jahresleistung des Solarturms bei einer bestimmten Positionierung der Heliostate wird das in Kapitel 2 beschriebene Raytracing-Modell verwendet. Zur Berechnung der optimalen Positionen der Heliostate wird in dieser Arbeit das Prinzip des Genetischen Algorithmus herangezogen. [7]

3.1 Funktionsweise eines Genetischen Algorithmus

Die Funktionsweise eines Genetischen Algorithmus ist angelehnt an die biologische Evolution. Daher wird bei diesem Algorithmus oft auch von einem evolutionären Algorithmus gesprochen. Das Prinzip ist, dass man aus einer Generation, die aus einer Population mit einer Menge von Individuen besteht, eine neue Generation züchtet, welche bessere Eigenschaften (Fitnesswert) besitzt und dieses iterativ wiederholt, um so möglichst nah an das globale Optimum zu gelangen. Um aus einer gegebenen Generation eine neue, bessere zu erzeugen, werden aus dieser zufällig Individuen selektiert (Selektion), die einem bestimmten Gütekriterium am besten entsprechen. Deren Eigenschaften (Genome) werden miteinander kombiniert (Rekombination) und teilweise verändert (Mutation). Diese Selektion, Rekombination und Mutation wird so oft wiederholt, bis ein Stoppkriterium (Terminator) erfüllt ist, wie z.B. eine Verbesserung der Generation zur vorherigen, die einen bestimmten Grenzwert nicht erreicht.
Im folgenden wird nun beschrieben, wie dieses Verfahren auf das Problem der Optimierung eines Solarturmkraftwerkes angewendet wird. Dabei wird auch direkt Bezug auf die in der Software verwendeten Klassen genommen, die später genauer beschrieben werden.

3.2 Generation

Eine Generation besteht aus der fixen Anzahl n von Individuen.

$$solPowerPlantList = \{\text{solPowerPlant}_1 ... \text{solPowerPlant}_n\} \tag{6}$$

Für die Initialisierung des Algorithmus wird zunächst die erste Generation berechnet. Dabei kann ein Individuum entweder zufällig erstellt werden. Es kann aber auch über eine Datei, die die Koordinaten und Größen der Heliostaten beinhaltet, geladen werden. Die Position und Art des Turms, so wie andere feste Objekte wie Flüsse, Wege o.ä. sind in jedem Individuum gleich.

3.3 Individuum

Ein Individuum repräsentiert ein komplettes Solarturmkraftwerk (solPowerPlant), genauer gesagt, eine abgegrenzte Fläche auf der eine Menge von Objekten (solObject) platziert wird. An der Position (0,0) steht ein Solarturm (solTower). Dieser Turm hat entweder einen 360° (solTowerCircle) oder einen einseitigen (solTowerRecangle) Receiver und gegebene Parameter wie Höhe, Breite, Tiefe des Turms, Längen- und Breitengrad des genauen Standortes und Maße des Receivers. Neben diesem Objekt gibt es die Heliostaten (solHeliostat). Diese besitzen ebenfalls Parameter wie Höhe, Breite und Tiefe, sowie die Position in Meter bezüglich des Nullpunktes, also dem Turm. Auch andere Objekte (solBarrier), die die Heliostaten beeinträchtigen können (Wege, Flüsse usw.) werden auf dieser Fläche positioniert, um bei der späteren Berechnung des Fitnesswerts und Neupositionierung der Heliostaten berücksichtigt werden zu können. Weitere Parameter, die für die Berechnung relevant sind, werden später in den jeweiligen Klassen genauer beschrieben.
Somit besteht ein Individuum aus einer Menge von Objekten:

$$\text{solPowerPlant}_i = \{\text{solObject}_{i,1}, ..., \text{solObject}_{i,v}\} \tag{7}$$

mit

$$\text{solObject}_{i,1}|\text{typeOf solTower} \tag{8}$$

$$\{\text{solObject}_{i,2}...\text{solObject}_{i,u}\}|\ \text{typeOf solBarrier} \tag{9}$$

$$\{\text{solObject}_{i,u+1}...\text{solObject}_{i,v}\}|\ \text{typeOf solHeliostat} \tag{10}$$

3.4 Genom

Ein Individuum besteht aus mehreren Genomen, in denen die Positionen der Heliostate gespeichert werden.

$$\text{solVector}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{11}$$

Die Genome können durch den Genetischen Algorithmus mittels Rekombination und Mutation verändert werden.

3.5 Fitnessfunktion

Eine Fitnessfunktion $f(\text{solPowerPlant})$ berechnet die Qualität eines Individuums. Es sind unterschiedliche Fitnessfunktionen denkbar wie beispielsweise die Effizienz, die eingefangene Strahlungsleistung oder die Stromerzeugungskosten. In der vorliegenden Arbeit wird die Qualität eines Individuums gegeben durch die Leistung in kWh/a, die ein

Solarturm mit m Heliosteten und deren Positionierung über einen Zeitraum von einem Jahr erreichen kann. Dieser Fitnesswert wird mit dem Raytracing Modell (solTowerModelRayTrace) aus Kapitel 2 berechnet. Ein Individuum solPowerPlant_j erbringt eine höhere Leistung als solPowerPlant_i, falls gilt:

$$f(\text{solPowerPlant}_i) < f(\text{solPowerPlant}_j) \tag{12}$$

3.6 Neue Generation

Um eine neue Generation zu generieren, werden dafür die Individuen aus der aktuellen Generation gezüchtet, indem zwei Individuen (solPowerPlant_W und solPowerPlant_M) der aktuellen Generation (solPowerPlantList) als Elternpaar selektiert werden und aus diesen ein neuer Nachkomme gebildet wird. Der Nachkomme wird jetzt eventuell noch mutiert und der neuen Generation hinzugefügt. Die Selektion, Rekombination und Mutation wird so oft wiederholt, bis die neue Generation genauso viele Individuen besitzt wie die alte.
Dabei werden die besten ℓ Individuen der alten Generation immer auch unverändert in die neue Generation übertragen.

Selektion Die Selektion erfolgt zufällig durch die Roulette Wheel-Methode. Abhängig von der relativen Qualität, d.h. im Verhältnis zur Gesamtleistung $f(\text{solPowerPlantList})$ der Generation, siehe Abbildung 9. Die Gesamtleistung berechnet sich über die Summe der einzelnen Individuen der Generation.

$$f(\text{solPowerPlantList}) = \sum_{i=1}^{n} f(\text{solPowerPlant}_i) \tag{13}$$

Für die Roulette Wheel-Methode werden die Bereiche definiert, in denen ein Individuum ausgewählt wird.

$$L = \{(b_1, e_1), (b_2, e_2), ..., (b_n, e_n)\} \tag{14}$$

Der Bereich von b_i bis e_i steht dabei für das i-te Individuum.

$$b_i = \begin{cases} 0, & \text{wenn } i = 1 \\ e_{i-1}, & \text{sonst} \end{cases}, \; e_i = b_i + \frac{f(\text{solPowerPlant}_i)}{f(\text{solPowerPlantList})} \tag{15}$$

Es wird eine Zufallszahl zwischen 0 und 1 gewählt.

$$r_W, r_M \in random(0, 1) \tag{16}$$

Nun können der Bereich, in dem der Zufallswert liegt und das entsprechende Individuum ausgewählt werden.

$$\text{solPowerPlant}_W = \text{solPowerPlant}_i \text{ mit } b_i \leq r_W \leq n_i \tag{17}$$

bzw.

$$\text{solPowerPlant}_M = \text{solPowerPlant}_j \text{ mit } b_j \leq r_M \leq n_j \tag{18}$$

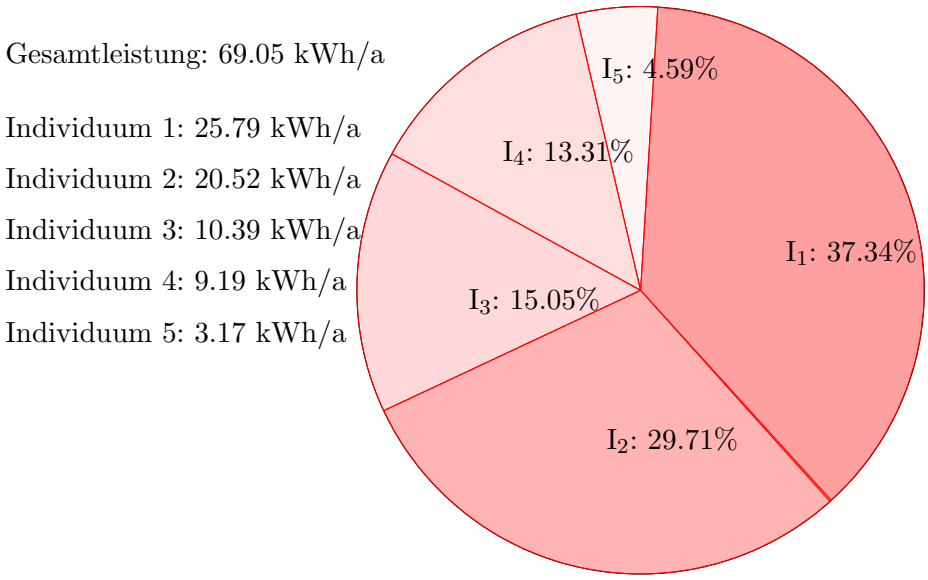
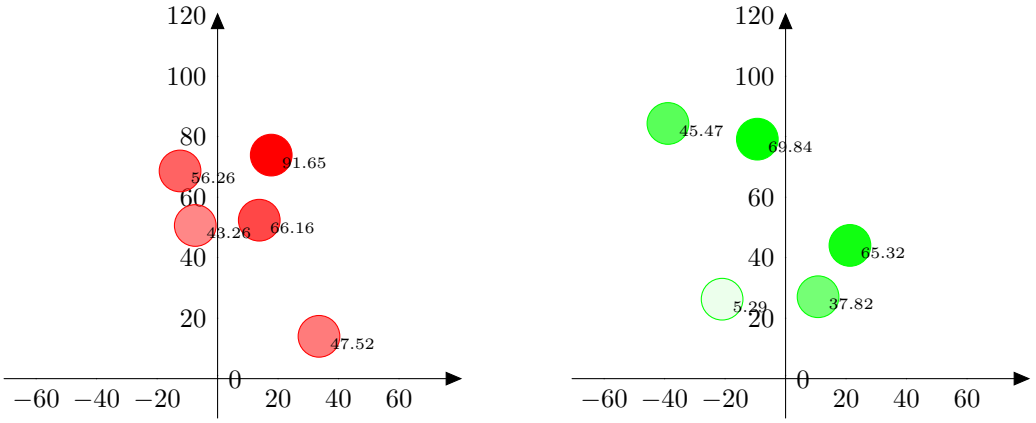


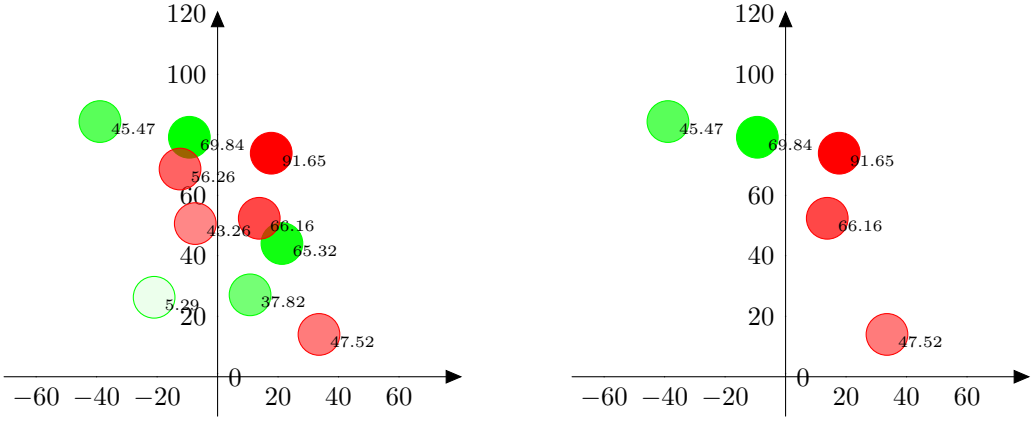
Abbildung 9: Roulette Wheel Methode zur Selektion eines Individuums
I_1 wird dabei mit einer Wahrscheinlichkeit von 37,34% ausgewählt, im Gegensatz zu I_5, welches nur in 4,59% der Fälle gewählt wird.

Rekombination Die Erzeugung der Nachkommen, also der Individuen der neuen Generation, erfolgt durch eine Rekombination der Elternteile, welche durch die Selektion ausgewählt wurden. Zunächst erhält jeder Heliostat beider Elternteile eine Wertigkeit, die sich aus den Verlusten in Kapitel 2 ergibt. Die Heliostate werden gemäß dieser Wertigkeit absteigend sortiert. Die einzelnen Heliostate werden nacheinander in das neue Individuum eingefügt. Hierbei werden jedoch nur die Heliostate eingefügt, die nicht mit den bereits eingefügten kollidieren, sodass nur gültige Individuen entstehen.
Im folgenden kleinen Beispiel, siehe Abbildungen 10 und 11, soll diese Rekombination mit Individuen gezeigt werden, die nur 5 Heliostate besitzen.



(a) solPowerPlant$_f$ (b) solPowerPlant$_m$

Abbildung 10: Selektion der beiden Elternteile solPowerPlant$_f$ und solPowerPlant$_m$



(a) Kombination aller Heliostate (b) Gültiger Nachkomme

Abbildung 11: Auswahl der 5 besten und gültigen Genome

Mutation Bevor ein Nachkomme nach der Selektion und Rekombination in die neue Generation hinzugefügt wird, werden einzelne Genome verändert. Diese Mutation führt ähnlich wie in der Natur zu einer Artenvielfalt in der neuen Generation, die sonst keine zu generierenden Individuen hervorbringt.

Eine Mutation ist eine Verschiebung eines Heliostaten auf dem Feld. Ein zu lösendes Problem bei der Mutation besteht darin, dass das Individuum gültig bleiben muss. Die Heliostate dürfen nach einer Mutation nicht kollidieren. Um dies zu verhindern, wird zunächst ein maximaler Radius angegeben, in dem der Heliostat verschoben werden darf. Jetzt wird der nächste Nachbar des zu mutierenden Heliostaten gesucht und die genaue Entfernung berechnet. Unterschreitet die Distanz den maximalen Radius, wird er auf die

Entfernung zu diesem Nachbarn reduziert, siehe Abbildung 12.

$$\text{distance} = \min(\text{maxDistance}, \text{neighborDistance}) \tag{19}$$

Jetzt kann ein zufälliger Winkel zwischen 0 und 2π

$$\text{randomAngle} = random(0, 1) \cdot 2\pi \tag{20}$$

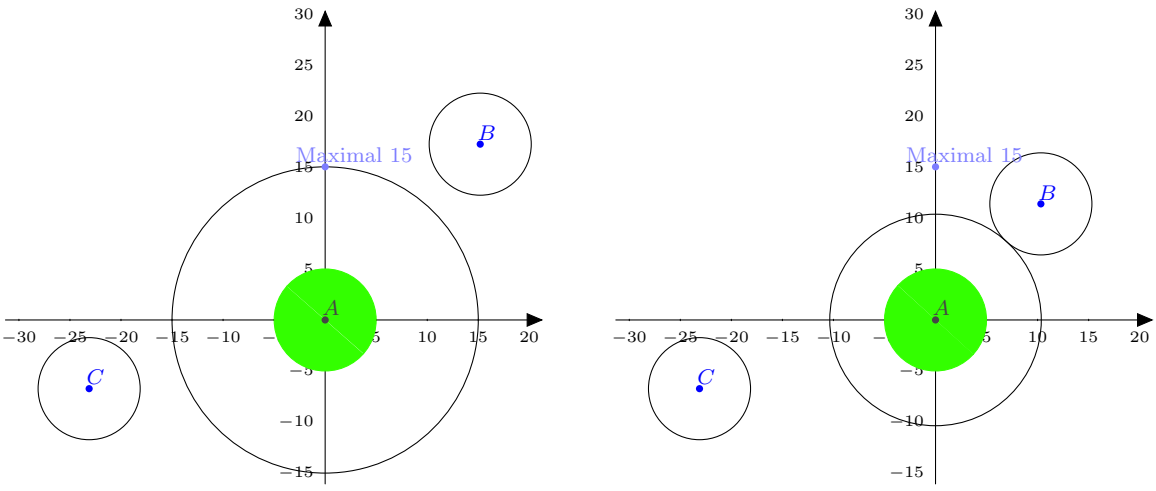
und ein Abstand zwischen 0m und der maximal zulässiger Distanz erzeugt werden,

$$\text{randomDistance} = random(0, 1) \cdot \text{distance} \tag{21}$$

und der Heliostat kann neu positioniert werden.

$$\tilde{x} = \text{randomDist} \cdot \cos(\text{randomAngle}) + x \tag{22}$$

$$\tilde{y} = \text{randomDist} \cdot \sin(\text{randomAngle}) + y \tag{23}$$

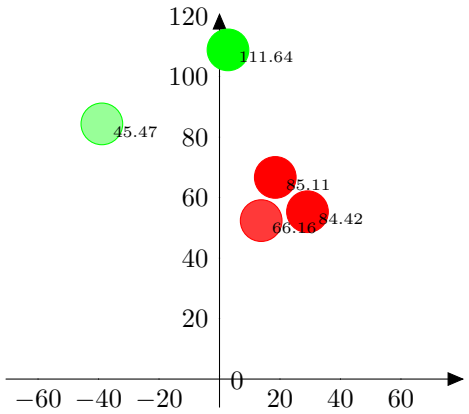


(a) Maximaler Abstand = maxDistance

(b) Maximaler Abstand = Abstand A zu B

Abbildung 12: In Szenario a) liegen alle Nachbarn zu A außerhalb der maximalen Distanz, in b) reduziert sich der maximale Abstand durch den Nachbarn B.

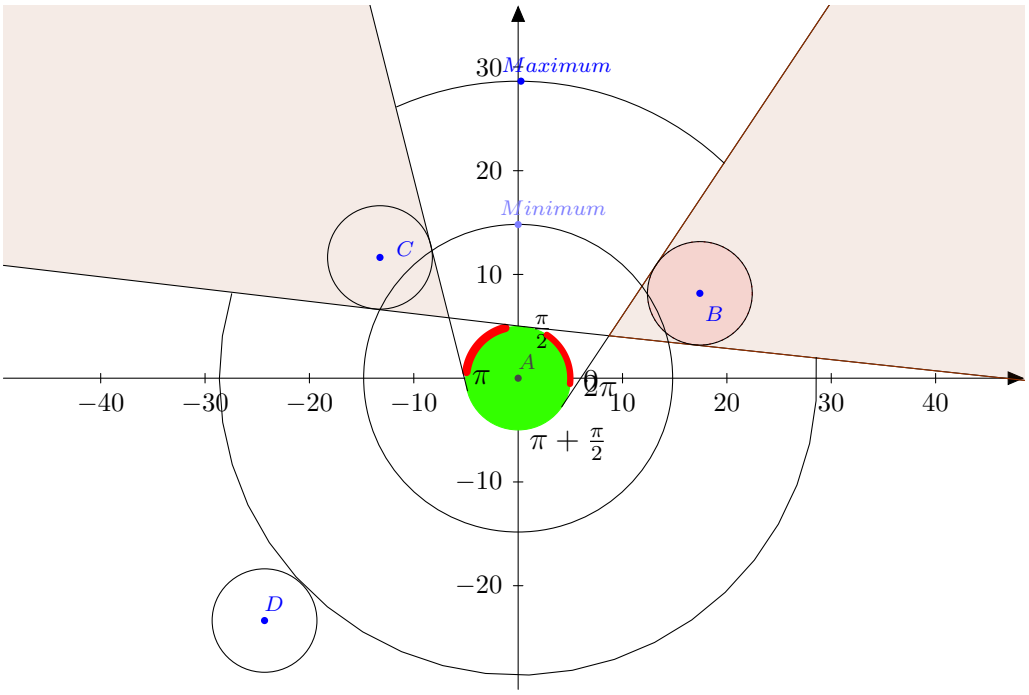
Damit wird verhindert, dass es zu einer Kollision mit Nachbar-Objekten kommen kann. Allerdings können sich auf diese Weise Gruppen von Objekten bilden, die sich gegenseitig blockieren und sich durch die Nähe zu den anderen Objekten nicht mehr mutieren lassen, siehe Abbildung 13.



(a) Gruppenbildung von Heliostaten

Abbildung 13: Die roten Heliostate lassen sich nicht weiter, bzw. nur minimal mutieren, da sie sich gegenseitig blockieren.

Das Problem ist, dass ein Nachbar, der sehr nahe bei dem zu mutierenden Heliostaten liegt, den kompletten Mutationsradius einschränkt. Es ist jedoch nur nötig, die Richtung, in welcher der Nachbar liegt, zu beschränken. Der Heliostat kann sich dann in die anderen Richtungen weiter frei bewegen, ohne Gefahr zu laufen, mit einem anderen Heliostaten zu kollidieren.
Zuerst wird daher ein minimaler Abstand definiert, welcher angibt, ob ein Nachbar zu nah an dem zu mutierenden Heliostaten liegt. Sollte dies der Fall sein, wird nicht der maximale Radius eingeschränkt, sondern der Winkelbereich, in dem die Mutation stattfinden kann, ohne den nahen Nachbarspiegel zu berühren. Auch bei jedem weiteren Nachbarn, der näher als dem minimalen Abstand entfernt ist, muss diese Einschränkung erfolgen. In Abbildung 14 wird so eine Situation mit 3 Nachbarspiegeln gezeigt.



(a) Winkelbereichs-Einschränkung

Abbildung 14: Zu mutierender Heliostat A und Nachbar B und C, welche den minimalen Abstand unterschreiten und den Winkelbereich einschränken. Nachbar D schränkt nur den maximalen Radius ein.

3.7 Test des Genetischen Algorithmus

Im weiteren Verlauf wird zum Testen der Effektivität des Genetischen Algorithmus nicht das in Kapitel 2 vorgestellt Raytracing Modell verwendet, sonder Test-Zielfunktionen (solTowerModelTest), deren Berechnung lediglich auf den x- und y-Positionen der Heliostaten basiert.
Die Qualität eines Individuums ist somit gegeben durch Funktionen $g(x, y)$ mit

$$f(\text{solPowerPlant}) = \sum_{i=1}^{m} g(solHeliostat_{i,x}, solHeliostat_{i,y}) \tag{24}$$

Mit der Test-Zielfunktion $g_1(x, y) = x + y$ werden oben rechts bessere Werte erreicht als unten links. In Abbildung 15 wird grafisch dargestellt, wie eine Optimierung unter Verwendung der Testfunktion aussehen könnte. Repräsentanten der Generationen sind jeweils die Individuen mit den besten Fitnesswerten. Ein Kreis steht hierbei für die Position eines Heliostat im Feld und die Farbe für die Qualität. In Abbildung 16 werden weitere Optimierungen von den Testfunktionen

$$g_2(x, y) = 500 - |x| + 400 - |400 - y| \tag{25}$$

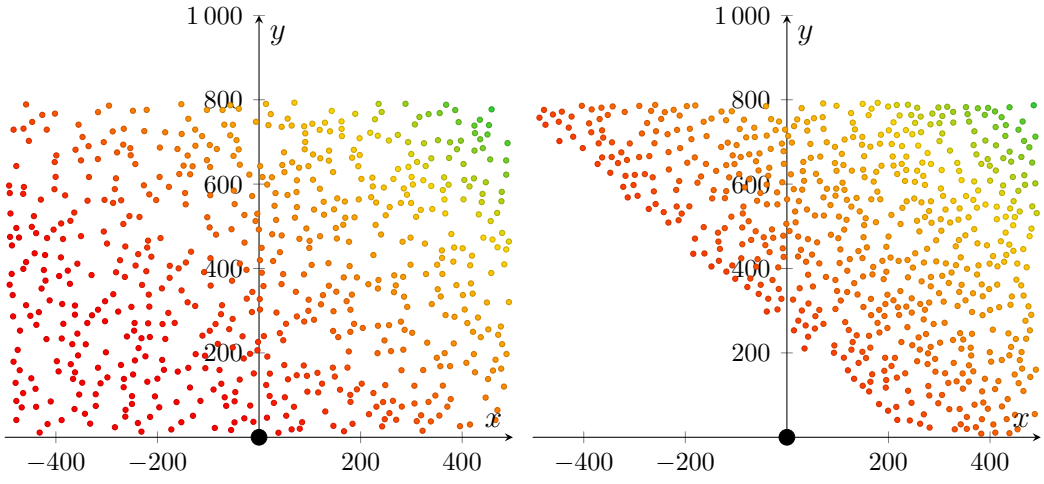
$$g_3(x, y) = 800 - y \tag{26}$$

$$g_4(x, y) = 400 - |400 - y| \tag{27}$$

$$g_5(x, y) = 500 - |x| \tag{28}$$

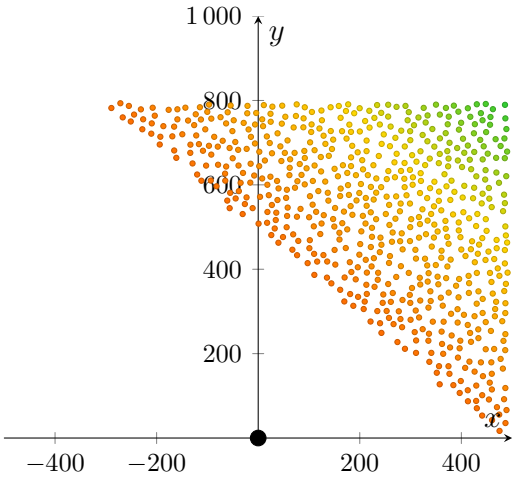
$$g_6(x, y) = \sqrt{x^2 + (400 - y)^2} \tag{29}$$

dargestellt.

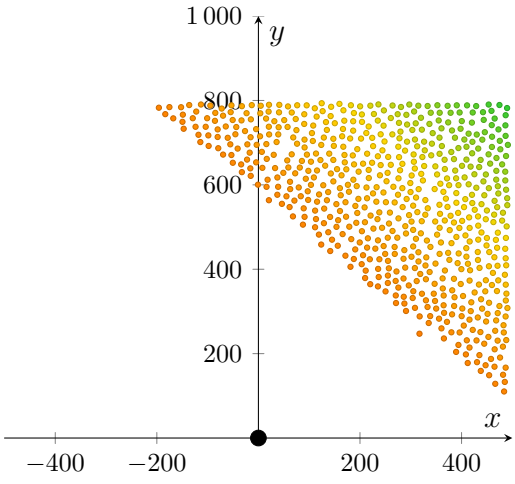


(a) Zufällige Positionierung

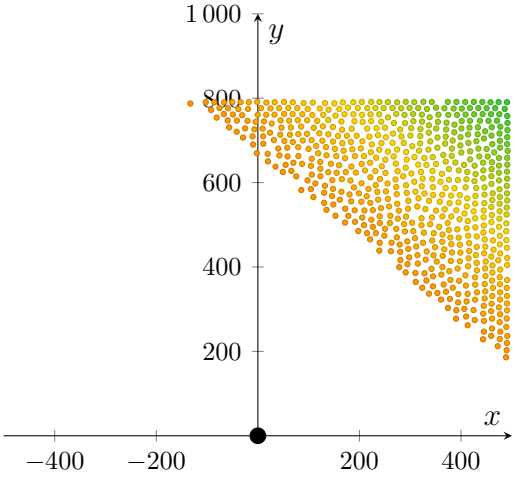
(b) Optimierungsschritt 2



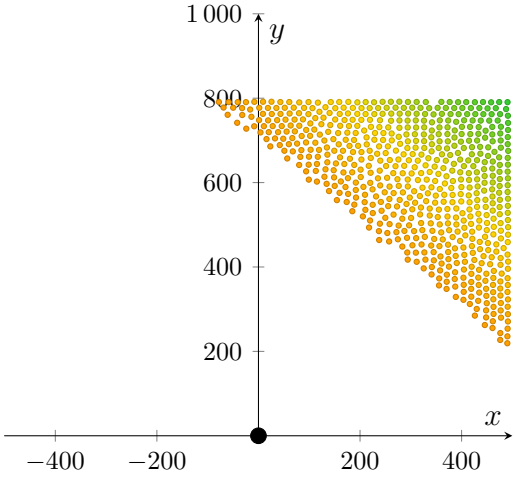
(c) Optimierungsschritt 4



(d) Optimierungsschritt 8

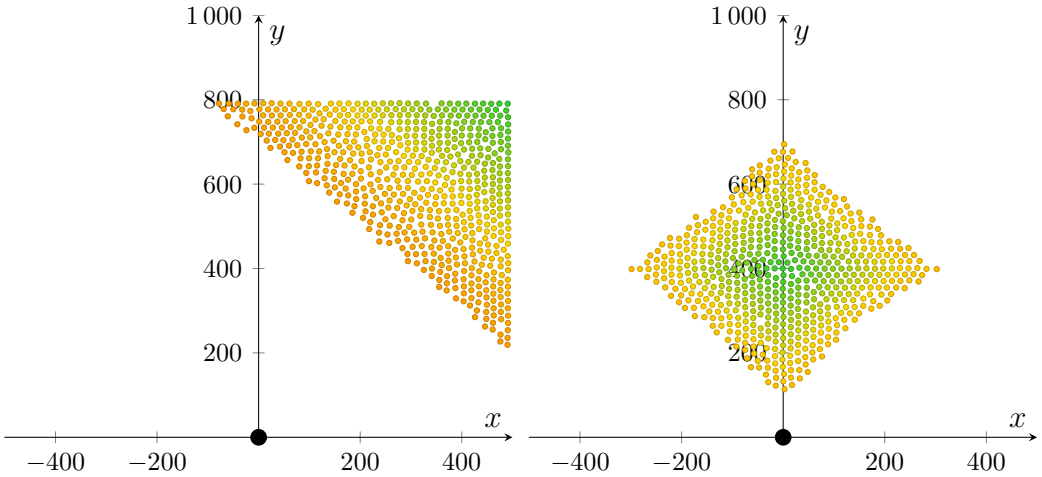


(e) Optimierungsschritt 50



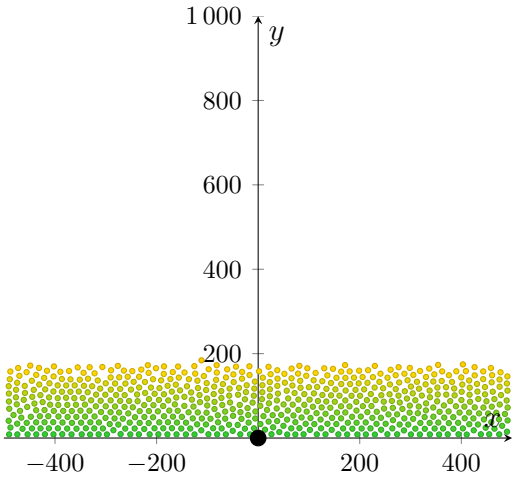
(f) Optimierungsschritt 100

Abbildung 15: Optimierungsschritte unter Verwendung der Testfunktion mit 624 Heliostaten und eine Populationsgröße von 20 Individuen

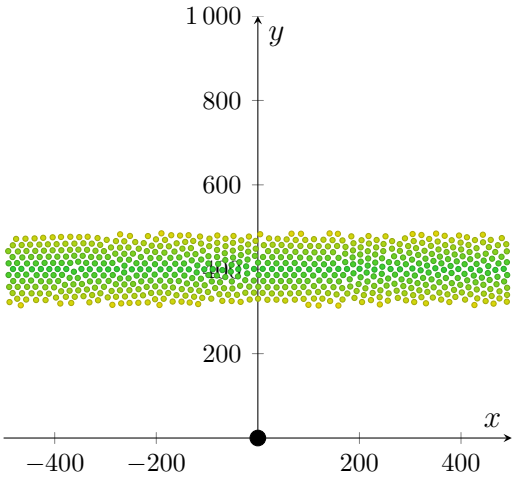


(a) $x + y$

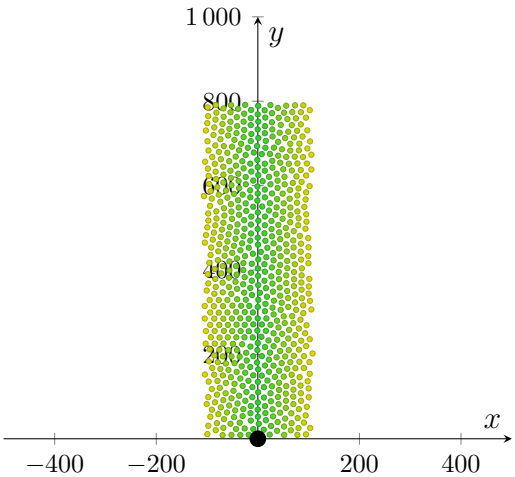
(b) $500 - |x| + 400 - |400 - y|$



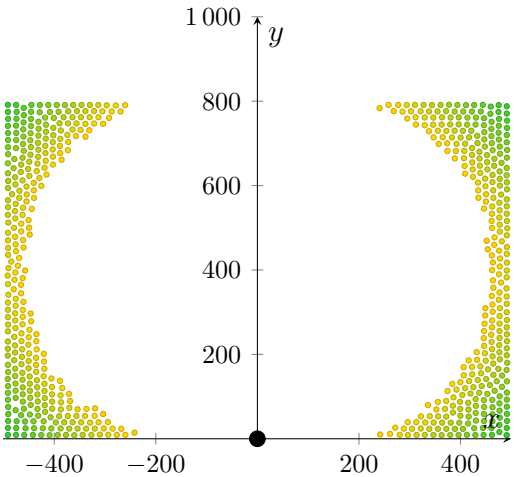
(c) $800 - y$



(d) $400 - |400 - y|$



(e) $500 - |x|$



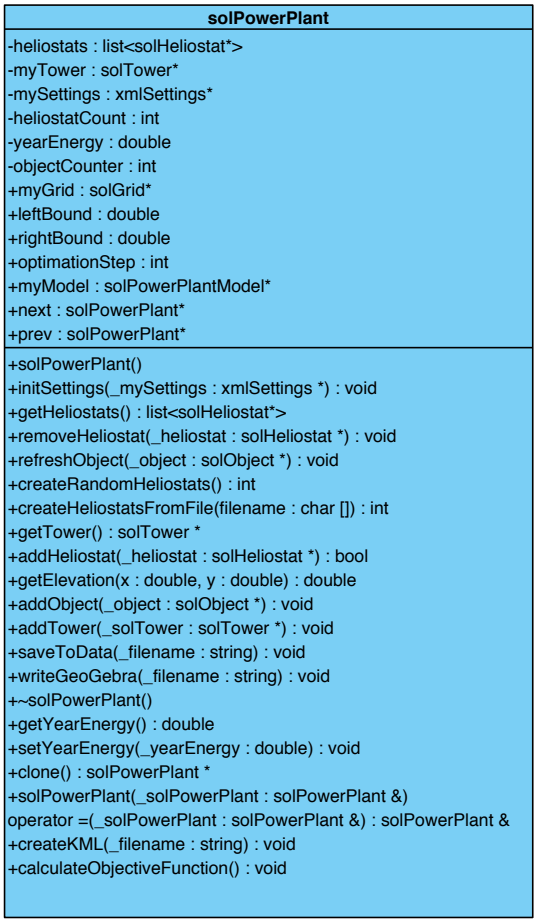
(f) $\sqrt{x^2 + (400 - y)^2}$

Abbildung 16: Optimierung von verschiedenen Testfunktionen

4 Software - Beschreibung der Klassen

Die Software ist mit C++ objektorientiert entwickelt worden. Sie bietet dadurch eine einfache Möglichkeit der Anpassung und Erweiterung. Im Folgenden sollen nun die einzelnen Klassen beschrieben werden.
Alle einzustellenden Parameter werden in einer xml-Datei verwaltet und ggf. bei der Initialisierung des entsprechenden Objektes übergeben. Verwendet wird dazu ein Objekt der xmlSettings Klasse, welche die Parameter aus der xml-Datei liest.

4.1 Individuum



Die Klasse solPowerPlantbildet das Individuum für den Genetischen Algorithmus.

initSettings initialisiert das Objekt und setzt alle Parameter. Hier wird auch ein solGrid Objekt und ein durch die xml-Settings definiertes solPowerPlantModel Objekt erstellt.

getHeliostats gibt eine Liste von Zeigern auf die bereits durch addHeliostat eingefügten Heliostaten zurück.

createRandomHeliostats erstellt eine durch die xmlSettings definierte Anzahl [HELIOSTATCOUNT] von zufällig positionierten Heliostaten. Die möglichen Positionen werden durch die Feld-Grenzen [FIELDMAXX, FIELDMAXY, FIELDMINX, FIELDMINY] eingeschränkt.

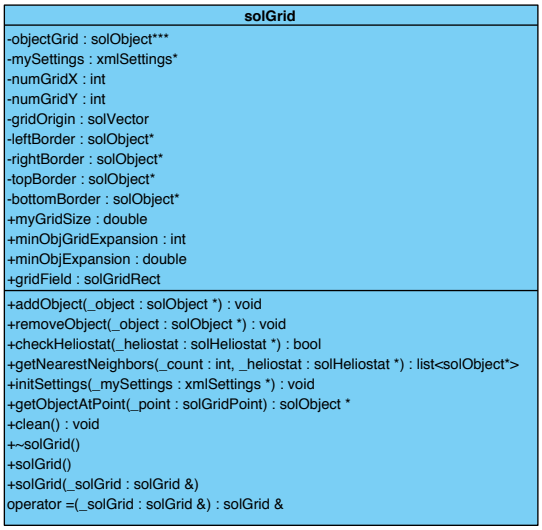
createHeliostatsFromFile Über dat-Dateien [DATAFILES] können vordefinierte Heliostate geladen werden. Es werden jedoch maximal die vorgegebene Anzahl von Heliostaten geladen oder, falls sie geringer ist, durch zufällige ergänzt.

saveToData speichert die Positionen und Abmessungen der hinzugefügten Heliostaten in eine dat-Datei.

addHeliostat prüft zunächst, ob das Einfügen des Heliostaten zu keiner Kollision mit anderen Objekten führt und fügt bei gültiger Prüfung das Heliostat Objekt in eine Liste (heliostats) und in das Datenmodell solGrid ein.

calculateObjectiveFunction berechnet den Wert der Zielfunktion, gegeben durch das über die xmlSettings gesetzte Modell myModel [OBJECTIVE_FUNCTION].

4.2 Gitter



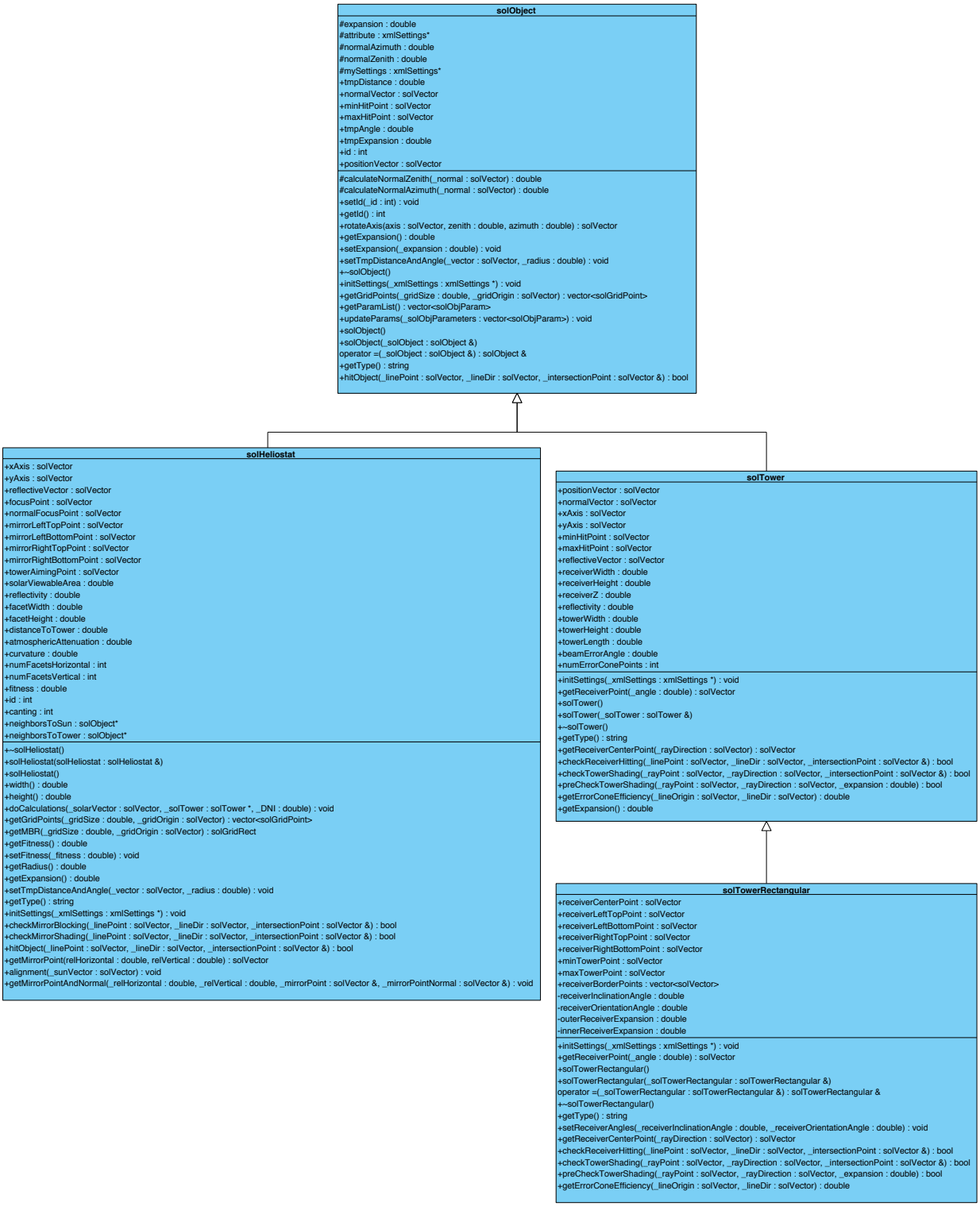
Neben einem list-Objekt (heliostats) werden die einzelnen Objekte zusätzlich in einem zweidimensionalen Array mit Listen von Zeiger auf Objekte vom Typ solObjekt gespeichert. Die Indizes [0][0] in diesem 2D-Array stehen dabei für die linke untere Ecke des Solar-Feldes und [maxWidth][maxHeight] für die rechte obere Ecke. Jeder Index in dem Gitter lässt sich auf einen Punkt auf dem Feld projizieren. Dabei können die entsprechenden Koordinaten anhand einer einstellbaren Rastergröße [GRIDSIZE] und der Abmaßen des Feldes berechnet werden. Welche Zeiger in dem Gitter auf welches Objekt zeigt ist abhängig von der realen Position der Objekte in dem Feld. Über das Feld wird

also ein Raster gelegt, bei dem jeder Rasterpunkt für einen Zeiger im Gitter steht. Die Rasterpunkte und damit die entsprechenden Zeiger in dem Gitter, die von dem minimal umgebenden Rechteck eines Objektes überdeckt werden, bekommen die Adresse dieses Objekts. Da diese Rechtecke nicht immer genau auf den Gitter-Punkten liegen, werden sie soweit vergrößert, dass sie genau auf den Punkten liegen.

checkHeliostat überprüft, ob die Zeiger der Gitter-Punkte, die auf das einzufügende Heliostat zeigen soll, bereits auf ein anderes Objekt zeigen. Sollte dies der Fall sein, wird die genaue Distanz zu diesem Objekt berechnet und bei einer Kollision der Wert $false$ zurückgegeben und so ein Einfügen verhindert. Anderenfalls wird mit $true$ das Einfügen erlaubt.

getNearestNeighbors gibt $\_count$ Objekte zurück, die dem Heliostat $\_heliostat$ am nächsten liegen. Dazu wird zunächst im Gitter das minimal umgebende Rechteck zu diesem Heliostat gesucht. Alle Gitter-Punkte, die im Abstand $minObjGridExpansion$, welcher der minimalen Größe aller im Feld vorhandener Objekte entspricht, von diesem Rechteck entfernt sind, werden nun überprüft, ob sie auf ein Objekt zeigen und werden ggf. in eine Ergebnisliste zwischengespeichert. Nun wird der Abstand der zu prüfenden Punkte um $minObjExpansion$ erhöht und es werden abermals die entsprechenden Punkte überprüft. Dies wird so oft wiederholt, bis die Ergebnisliste mindestens $\_count$ Elemente enthält. Diese werden nun der Distanz aufsteigend sortiert und die ersten $\_count$ Elemente zurückgegeben.

4.3 Solarturm-Objekte

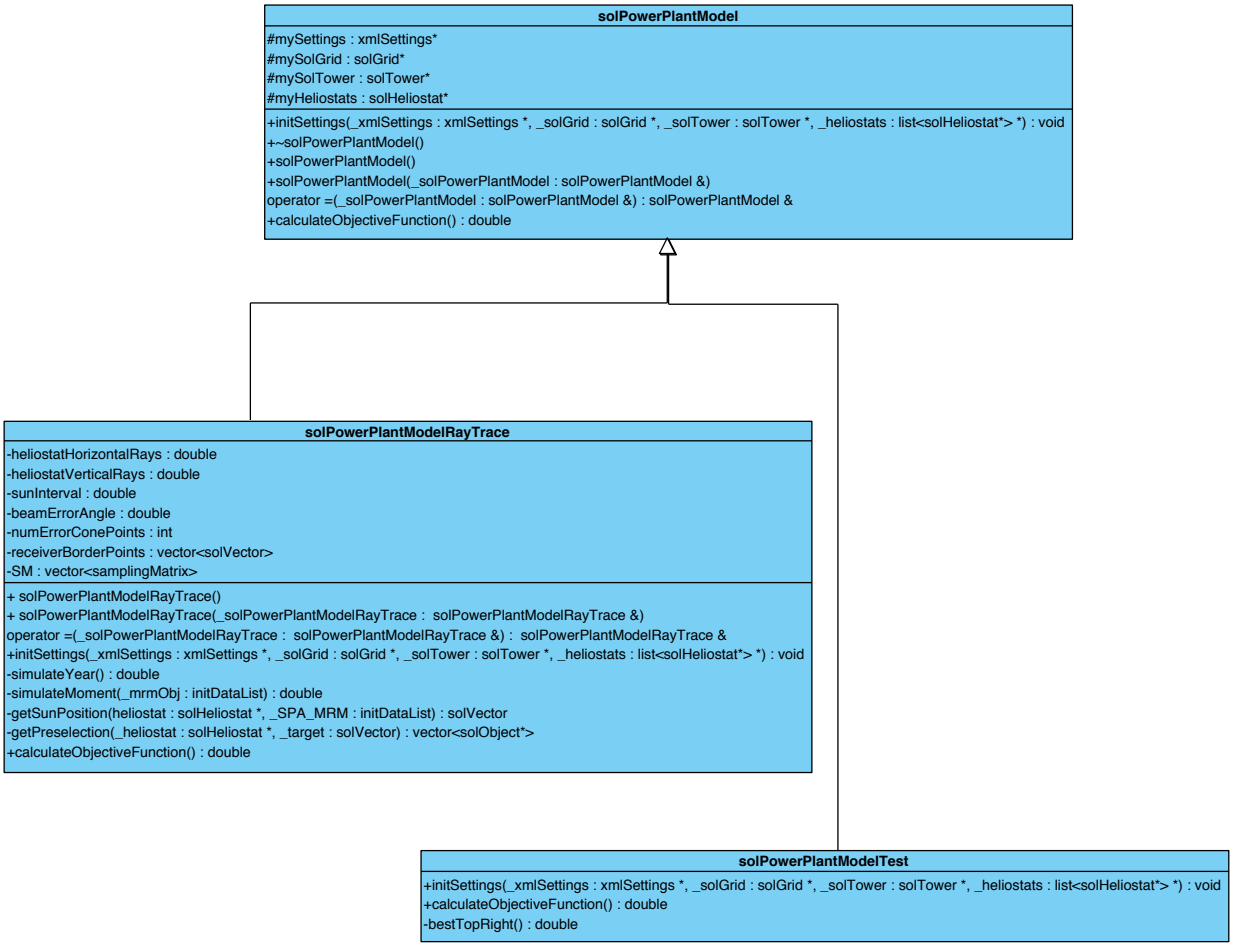


Jedes Objekt, das in das solPowerPlanteingefügt wird, ist vom Typ solObjectvererbt. Dies ist eine abstrakte Klasse, die als Template für alle Objekte dient und deren Methoden in den Kind-Klassen implementiert werden müssen. Dazu gehören unter anderem:

getGridPoints gibt die Punkte zurück, die durch das minimal umgebende Rechteck des jeweiligen Objekts überdeckt werden. Bei einem Heliostaten wären dies z.B. alle Punkte, die ein Quadrat der Breite von 2·getRadius(), also das minimal umgebende Rechteck, überdecken.

setTmpDistanceAndAngle bestimmt Abstand und Winkel zu einem Objekt mit dem Mittelpunkt (_solVector.x,_solVector.y) und dem Radius _radius und speichert diese Werte in die temporären Attribute tmpDistance und tmpAngle.

4.4 Modelle



Die abstrakte solPowerPlantModel Klasse dient als Template für ein Modell zur Berechnung einer Zielfunktion, welche den Fitnesswert des Individuums berechnet. Die Methode calculateObjectiveFunction() wird im Genetischen Algorithmus aufgerufen, um den

Fitnesswert des Individuums _solPowerPlantzu berechnen. Das Modell solPowerPlant-ModelRayTrace berechnet die Energie, die ein Solarturmkraftwerk über einen Zeitraum von einem Jahr produziert. Das Testmodell solTowerModelTest berechnet hierbei nicht den wirklichen Wert, sondern zum Verifizieren des Algorithmus die folgende Funktion:

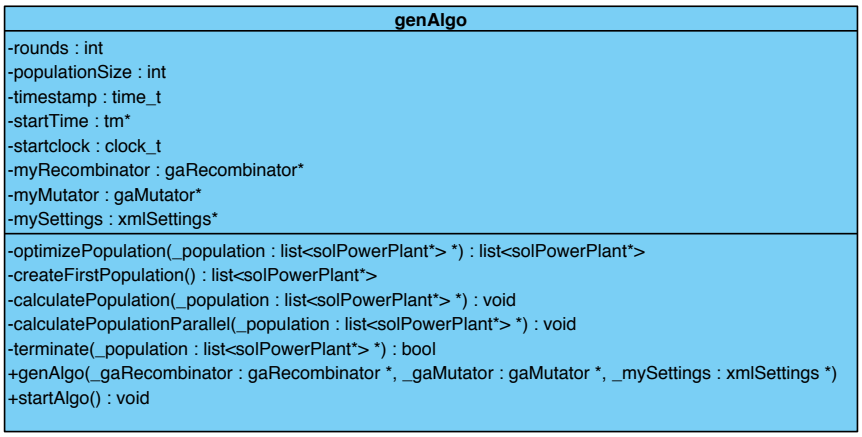
```
double solPowerPlantModelTest::calculateObjectiveFunction(){
        return bestTopRight();
}

double solTowerModelTest::bestTopRight() {
        double energy = 0.0;
        list<solHeliostat*> helList = mySolPowerPlant->getHeliostats();
        list<solHeliostat*>::iterator it;

        for(it = helList.begin();it!=helList.end();++it){
                solHeliostat *h = *it;
                h->setFitness(h->positionVector.x + h->positionVector.y);
                energy+= h->getFitness();
        }
        return energy;
}
```

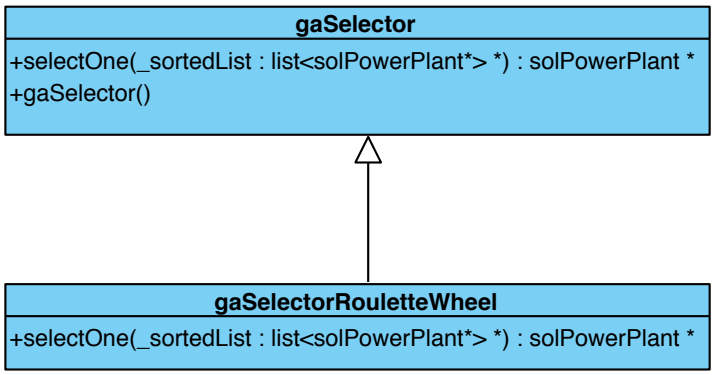
Es wird also die Position eines Heliostaten auf dem Feld betrachtet und der Fitnesswert ist die Summe der x- und y-Koordinate aller Heliostate. Je mehr Heliostate sich in der rechten oberen Ecke befinden, desto höher ist der Fitnesswert.
Die Klasse solTowerModelRayTrace basiert auf dem in Kapitel 2 eingeführten Modell. Dabei wird für eine Anzahl von Tagen [YEARPOINTS] und Messpunkten pro Tag [DAY-POINTS] zunächst eine Tabelle mit Daten wie Sonnenstand, Sonnenaufgang, Sonnenuntergang usw. für die Längen- und Breitengerade des zu betrachtenden Solarturms berechnet. Auf Grundlage dieser Daten und den Positionen der einzelnen Heliostate wird nun berechnet, wie viele Sonnenstrahlen von der Sonne über die Spiegel der Heliostaten auf den Receiver treffen. Dabei wird berücksichtigt, dass andere Objekte in dem Feld die Strahlen blockieren können. Durch Einführung des solGrids konnte diese Prüfung deutlich verbessert werden, da in dem Raster leicht herauszufinden ist, welche Objekte für die Prüfung effektiv in Frage kommen.

4.5 Genetischer Algorithmus



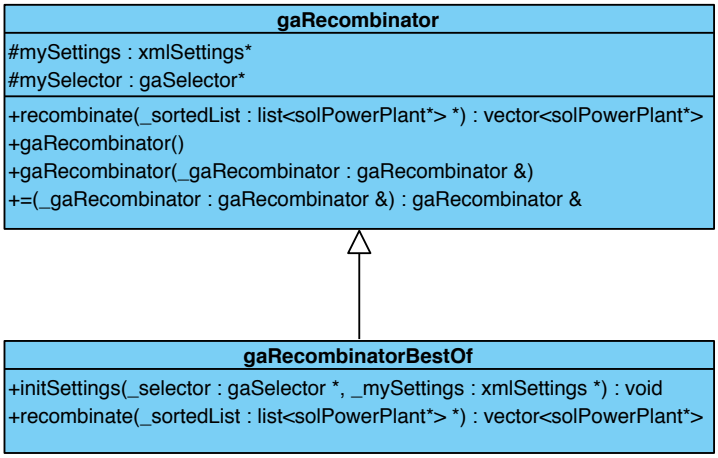
Diese Klasse beschreibt den kompletten Ablauf des Genetischen Algorithmus. Er startet mit der Methode startAlgo(). Dort wird zunächst mit createFirstPopulation() eine erste Generation anhand der Parameter aus mySettings erstellt. Man erhält eine Liste von Objekten des Typs solPowerPlantvon der Größe [POPULATIONSIZE]. Diese solPowerPlant's enthalten genau [HELIOSTATCOUNT] Heliostaten, entweder aus einer vorgegebenen dat-Datei oder zufällig erstellt. Über eine Schleife, die nach einer definierten Bedingung terminiert, wird die Population mit optimizePopulation() optimiert. Dazu wird zunächst eine Liste von Nachkommen mit dem gaRecombinator erstellt, siehe 4.7. Dies wird so oft wiederholt, bis die neue Population die entsprechende Anzahl von Individuen erreicht hat. Die Nachkommen werden nun durch den gaMutator verändert, siehe 4.8. Die neue Population wird zurückgegeben und mit calculatePopulation neu berechnet. Dabei wird das entsprechende solTowerModel verwendet, welches innerhalb des solPowerPlant instantiiert wird. Der Ablauf des Algorithmus und der Optimierungsschritt wird in Abbildung 17 und 18 gezeigt.

4.6 Selektor



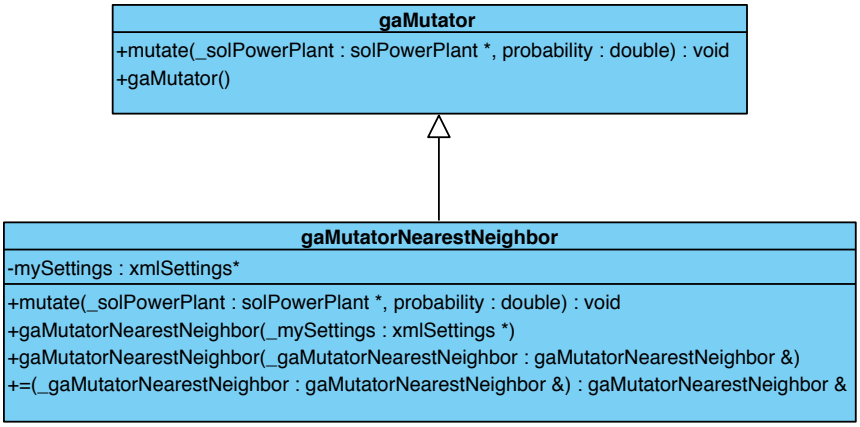
Der hier verwendete gaSelector gaSelectorRouletteWheel wählt, wie in 3.6 beschrieben ein Individuum zufällig aus, wobei Individuen mit höherem Fitnesswert wahrscheinlicher ausgewählt werden, als welche mit schlechterem.

4.7 Rekombinator



Der gaRecombinatorBestOf kombiniert zwei Individuen, die mit dem gaSelector mySelector ermittelt wurden zu einem Individuum, welches die besten und gültigen Heliostate beinhaltet, siehe 3.6

4.8 Mutator



Der gaMutatorNearestNeighbor mutiert mit einer Wahrscheinlichkeit _probability einen Heliostaten des Individuums _solPowerPlantindem zufällig eine neue Position gewählt wird, welche in einem gültigen Bereich zum nächsten Nachbarn liegt, siehe 3.6

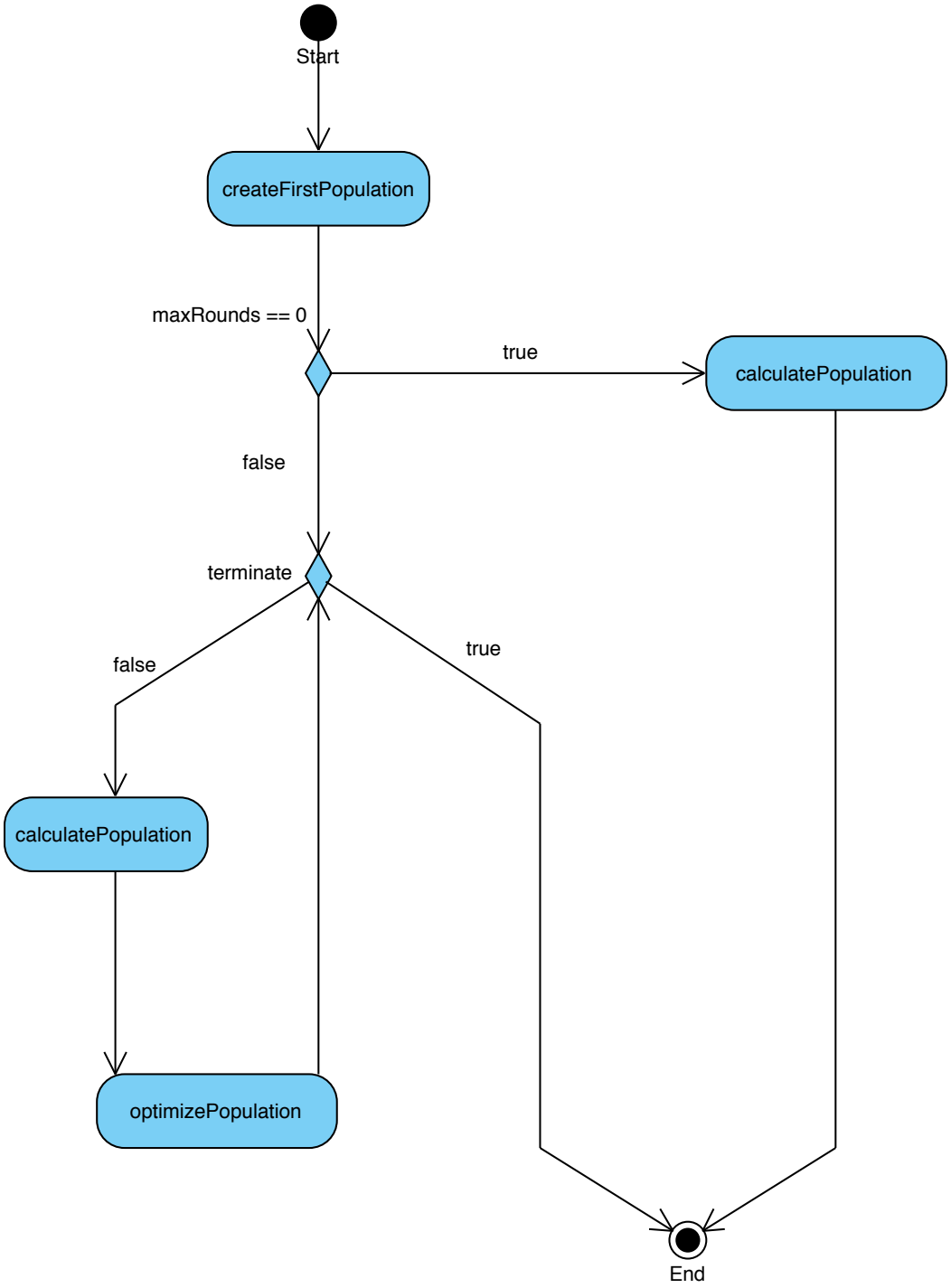


Abbildung 17: Ablauf des Genetischen Algorithmus

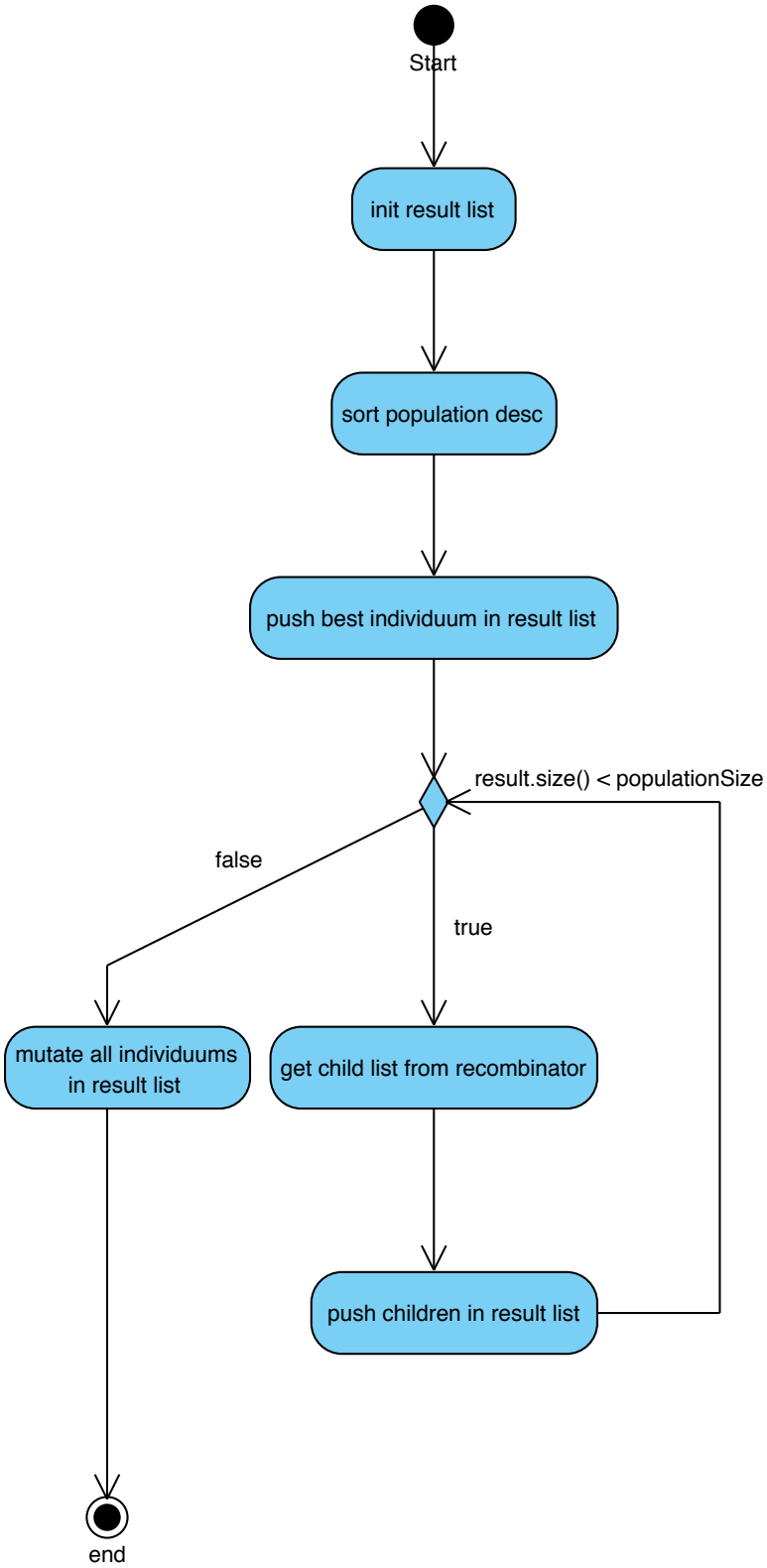


Abbildung 18: Ablauf eines Optimierungsschritts

5 Anwendung des Genetischen Algorithmus

5.1 Test der Gitterweite

Die Abstände der Rasterpunkte lässt sich beliebig einstellen. Es wurde ein Test erstellt, in dem die optimale Rastergröße in Abhängigkeit von der CPU-Zeit untersucht wurde. In Abbildung 19 wird das Verhältnis der Rastergröße zur Rechenzeit gezeigt. Bei der Untersuchung wurden Durchschnittswerte von mehreren Messungen, sowohl von einer Positionierung basierend auf dem Solarturm PS10, als auch einer zufälligen Belegung, genommen.

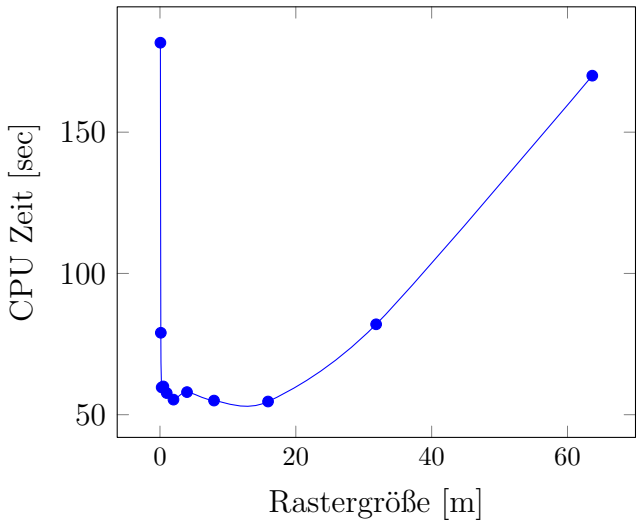


Abbildung 19: CPU Zeit in Abhängigkeit von der Rastergröße.

Es lässt sich zusammenfassend sagen, dass ein Wert zwischen 7 und 15 m die besten Ergebnisse liefert. Da die Heliostaten einen Durchmesser von ca. 15 m haben, kann man diesen Wert als Richtwert für die Rastergröße verwenden.

5.2 Parallelisierung

C++ bietet die Möglichkeit der Parallelisierung mit Hilfe der Bibliothek OpenMP [16]. Da die Fitnesswerte der einzelnen Individuen in einer Generation unabhängig voneinander berechnet werden können, ist hier die Parallelisierung sehr effizient. Der in Abbildung 20 dargestellte Graph zeigt die Zeitersparnis in Abhängigkeit zu den zur Verfügung stehenden Kerne auf der CPU.

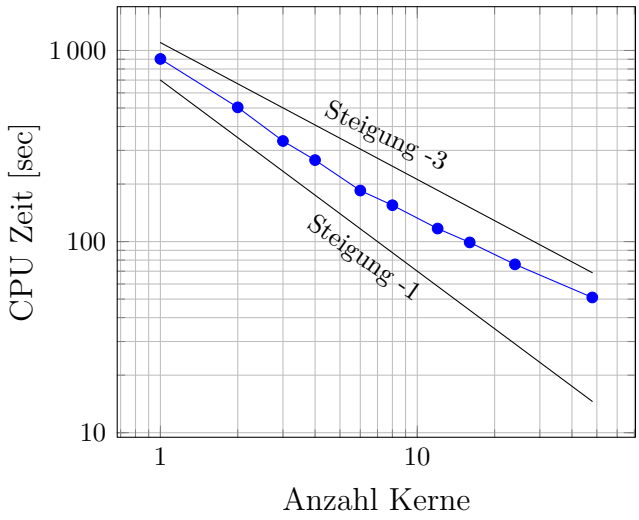
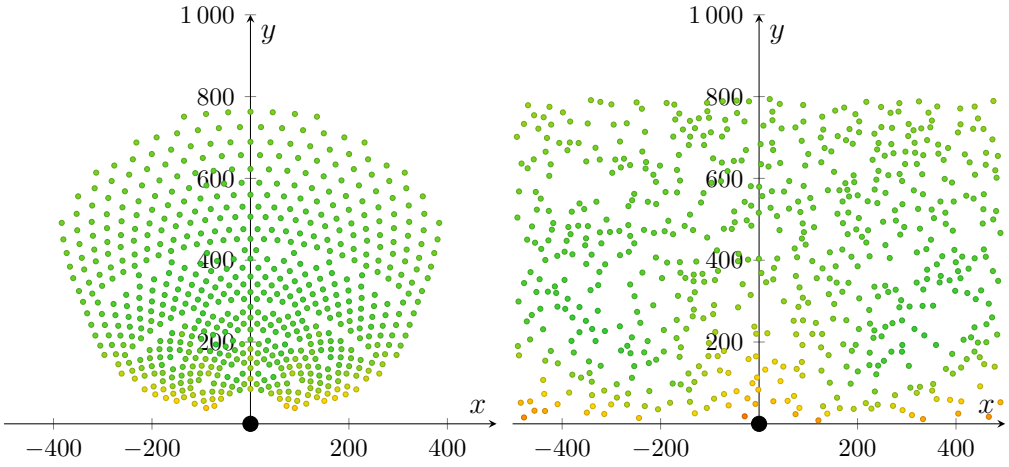


Abbildung 20: CPU Zeit in Abhängigkeit von der Anzahl der benutzten Kerne. Der Verlauf der Messpunkte liegt zwischen der unteren Linie mit Steigung -1 (dem theoretisch maximal möglichen Wert) und der oberen Linie mit Steigung -3.

Zu beachten ist hier allerdings, dass die optimale Ausnutzung aller Kerne nur dann gewährleistet ist, wenn die Anzahl der Individuen ein Teiler der verfügbaren Kerne oder kleiner ist. In einer zu parallelisierenden Schleife werden die Berechnungen auf die Kerne verteilt. Wenn die Anzahl der Berechnungen kleiner als die Anzahl der Kerne ist, werden alle in einem Durchlauf berechnet und zum etwa gleichen Zeitpunkt fertig. Müssen mehr Berechnungen gemacht werden, werden dementsprechend mehr Durchläufe ausgeführt, wobei jeder Durchlauf in etwa die selbe Zeit benötigt, unabhängig von den parallel laufenden Berechnungen. Daher ist es sinnvoll, eine Anzahl von Individuen zu wählen, so dass in jedem Durchlauf alle Kerne verwendet werden.
Die Parallelisierungsanalyse wurde auf einem System mit insgesamt 48 Kernen durchgeführt. Wie in Abbildung 20 zu erkennen, nähert sich die CPU Zeit bei maximalen Kernen nicht der Steigung von -1 an, sondern der Steigung von -3. Die Vermutung an dieser Stelle ist, dass auf dem von mehreren Wissenschaftlern gleichzeitig nutzbaren Server die maximale Anzahl von Kernen nicht zu jeder Zeit zur Verfügung steht und dadurch die CPU Zeit sich erhöht hat.

5.3 Modellgenauigkeit

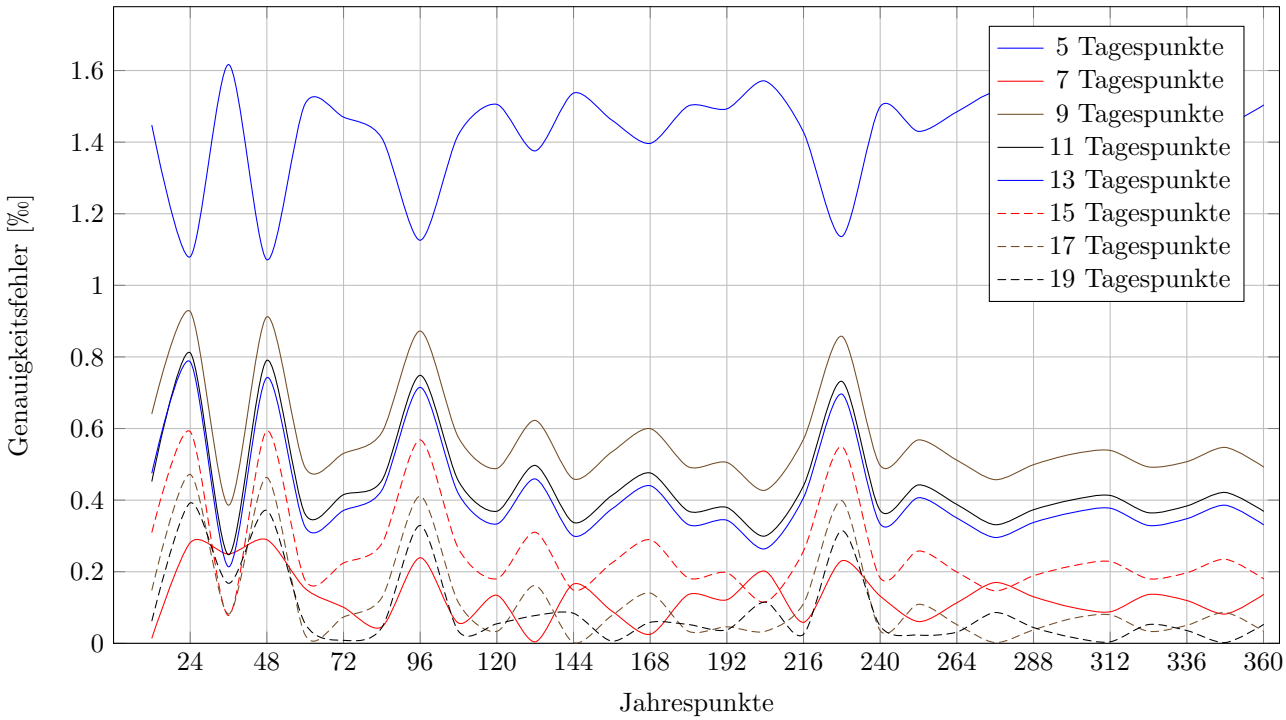
Das hier verwendete Modell zur Berechnung der Jahresleistung eines Solaturms berechnet zu bestimmten Momenten, gegeben durch eine Anzahl von Tagen im Jahr und Zeitpunkten pro Tag die am Receiver ankommende Energie und rechnet diese auf das Jahr hoch. Die Anzahl der Jahres- und Tagespunkte kann beliebig gewählt werden. Dabei werden die Jahrespunkte so verteilt, dass die Tage den selben Abstand haben, jedoch der 21. Juni, also der Tag der Sonnenwende genau ein Tag trifft, bzw. genau zwischen zwei Tagen liegt. Die Tagespunkte werden um die Mittagsstunde nach einer Gauß-Verteilung gesetzt. Je mehr Momente berechnet werden, desto genauer wird die Messung. Jedoch erhöht sich dadurch die Rechenzeit. Die folgende Analyse soll die Genauigkeit ermitteln, welche durch die Wahl der Jahres- und Tagespunkte erreicht werden kann. Bezugsgröße ist hierbei ein Wert, der mit 365 Jahrespunkten und 100 Tagespunkten berechnet wurde. Die Region, in dem der Solarturm steht und die zugehörigen meteorologischen Daten sind für diese Analyse ausschlaggebend, welche Ergebnisse erzielt werden. Als Vorlage wurde hier das Solarturmkraftwerk Planta Solar 10 (PS10) in Seville(Spanien) gewählt, siehe 21(a). In der ersten Messung (Abbildung 22) werden die Heliostaten Positionen dieses Kraftwerks verwendet und in der zweiten Messung (Abbildung 23) eine zufällige Positionierung.



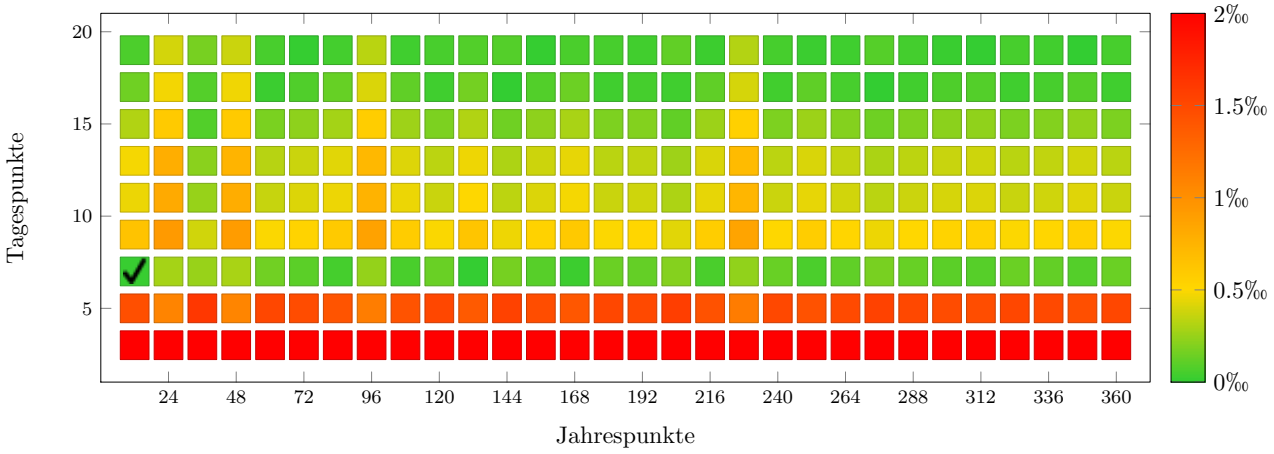
(a) PS10 Positionierung

(b) Zufällige Positionierung

Abbildung 21: Positionierung von Heliostaten

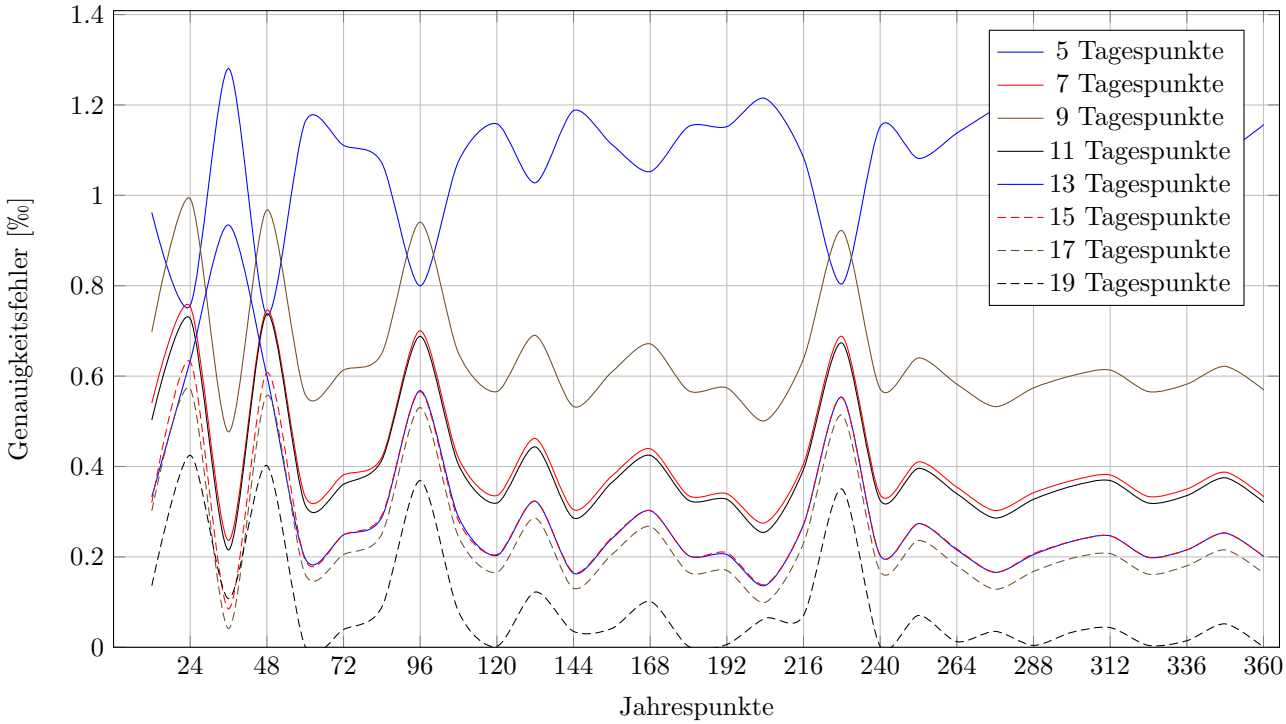


(a) Vergleich der Anzahl von Jahres- und Tagespunkten zur Genauigkeit

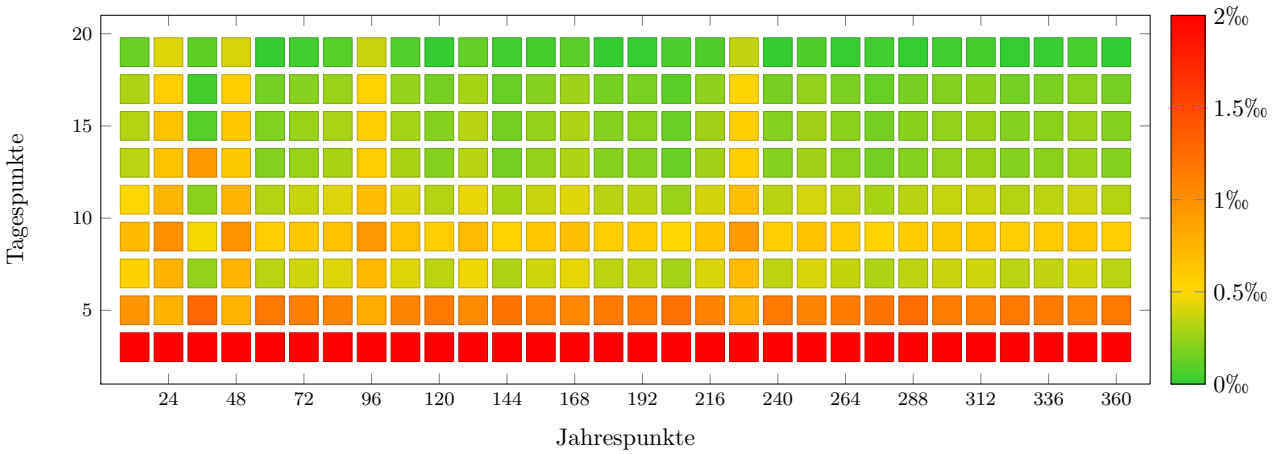


(b) Abweichung zum Optimalen Wert für Jahres-/Tagespunkt Kobinationen

Abbildung 22: Genauigkeitsanalyse für das Heliostatenfeld des PS10



(a) Vergleich der Anzahl von Jahres- und Tagespunkten zur Genauigkeit



(b) Abweichung zum Optimalen Wert für Jahres-/Tagespunkt Kobinationen

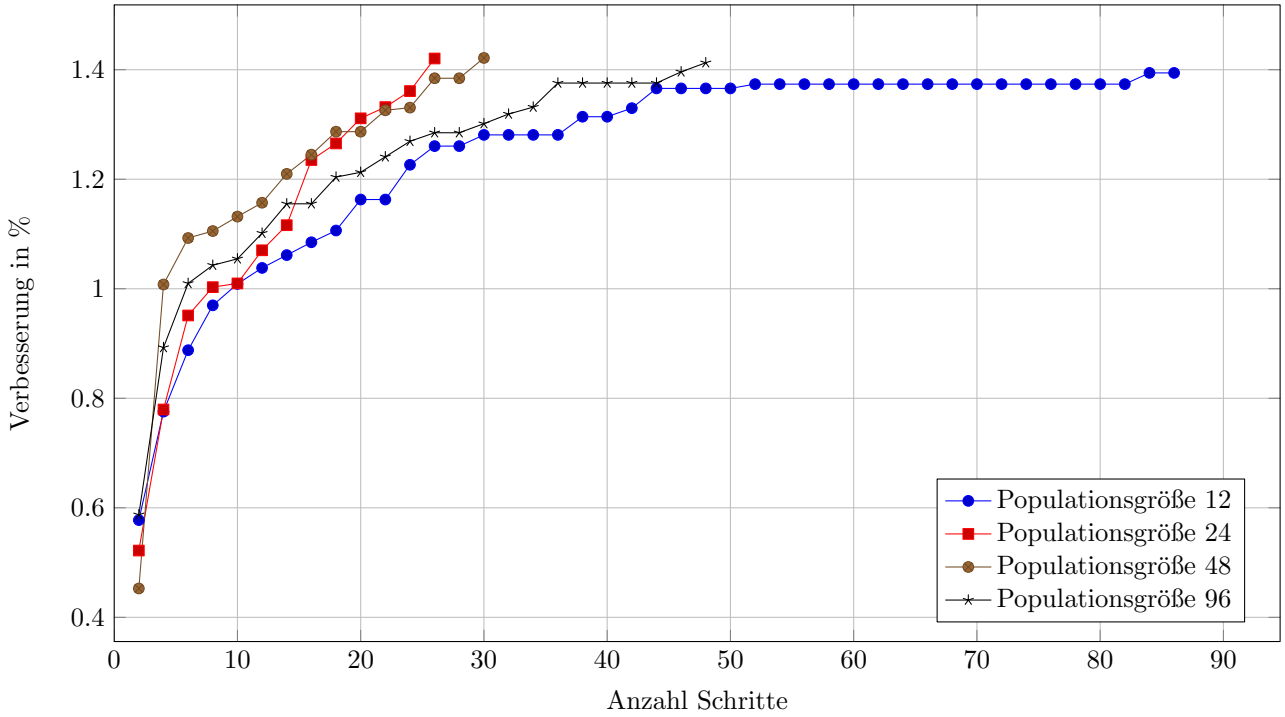
Abbildung 23: Genauigkeitsanalyse für ein zufällig belegtes Heliostatenfeld

Diese Analyse zeigt, dass mehr Jahrespunkte nicht unbedingt bessere Ergebnisse liefern. Daher ist es vorteilhafter, weniger Jahrespunkte zu nehmen, um die Rechenzeit zu verringern, jedoch einen, der genügend genaue Ergebnisse liefert. Die Anzahl der Tagespunkte sollte relativ hoch gewählt werden, da zu geringe Werte das Ergebnis verfälschen können. Sowohl die Werte für das gegebene Heliostatenfeld PS10, als auch für eine zufällige Belegung liefern für die Wahl von 36 Jahrespunkten und 7 Tagespunkten gute

Ergebnisse.

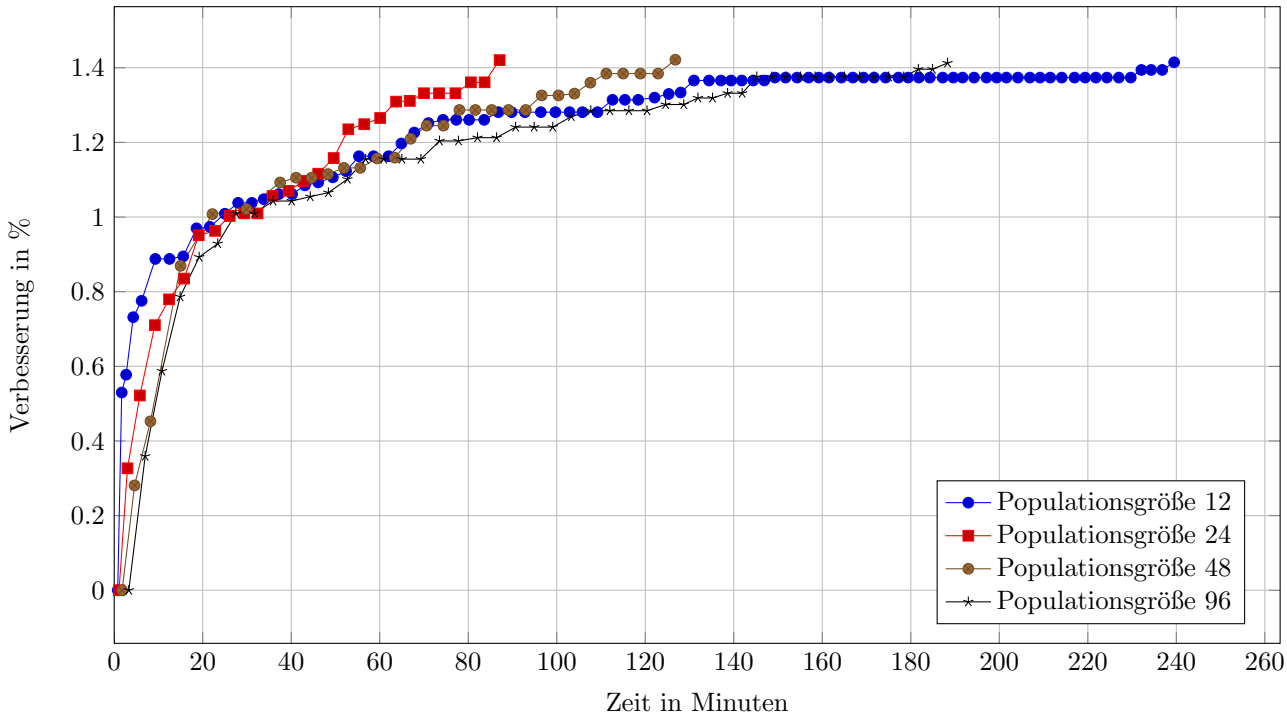
5.4 Populationsgröße

Je höher die Anzahl der Individuen in einer Population ist, desto höher ist die Wahrscheinlichkeit, dass in einer nächsten Generation Individuen erzeugt werden, die bessere Fitnesswerte besitzen. Ist die Zahl der Individuen jedoch zu hoch, gibt es in einer Generation auch viele Individuen mit schlechten Fitnesswert, welche durch die Selektion (siehe Kapitel 3.6) ausgewählt werden können. Dies führt dazu, dass die Verbesserung von Generation zu Generation langsamer verläuft. Auch die Rechenzeit nimmt zu, besonders wenn die Anzahl der Individuen höher ist als die der zur Verfügung stehenden Kerne (siehe Kapitel 5.2).
Der folgende Test soll zeigen, in wie vielen Schritten und in welcher Zeit der Algorithmus unter der Verwendung des Raytracing Modelles den Fitnesswert, also die Jahresleistung des Solarturmkraftwerks PS10 um 1,4% mit unterschiedlichen Populationsgrößen verbessern kann.

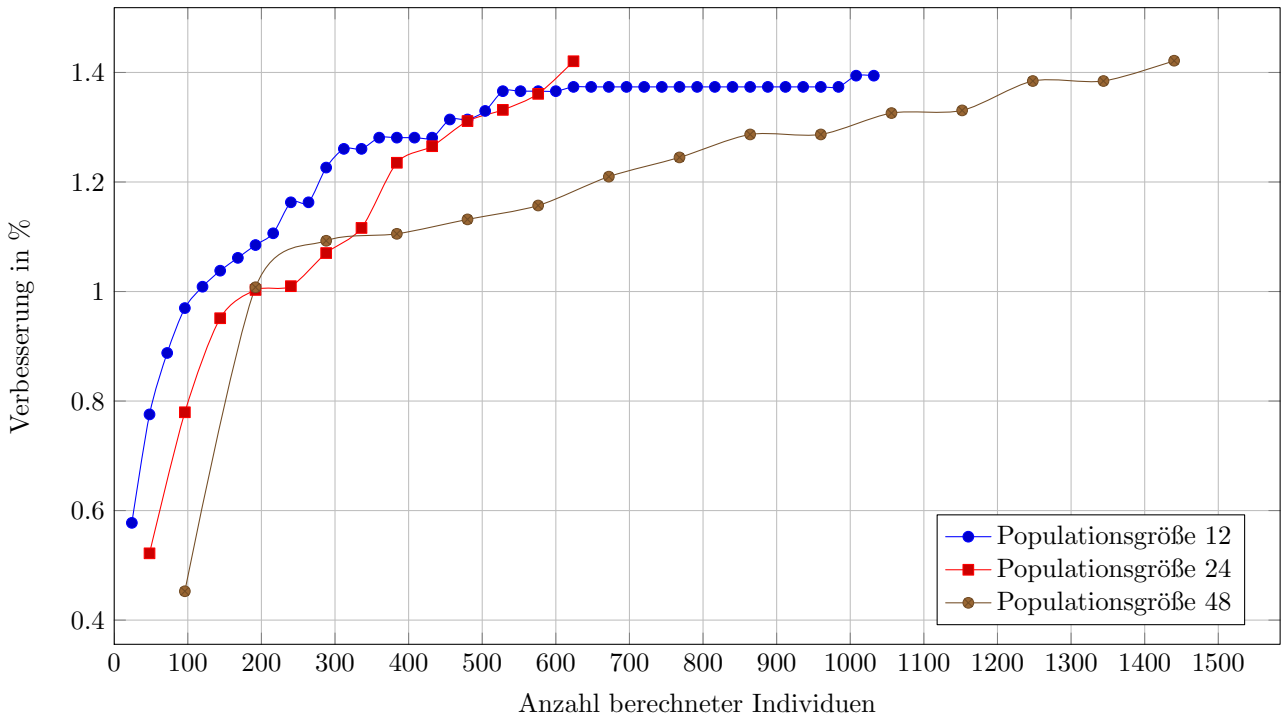


(a) Vergleich der benötigten Schritte, bis eine Verbesserung von 1,4% erreicht wird.

Abbildung 24: Test von verschiedenen Populationsgrößen



(b) Vergleich der benötigten Zeit, bis eine Verbesserung von 1,4% erreicht wird.



(c) Vergelich der zu berechnenden Individuen, bis eine Verbesserung von 1,4% erreicht wird

Abbildung 23: Test von verschiedenen Populationsgrößen

In Abbildung 23(a) ist zu sehen, dass eine Erhöhung der Populationsgröße nur bis zu

einer bestimmten Anzahl sinnvoll ist. Eine Anzahl von 96 Individuen pro Population erreicht die geforderte Verbesserung zwar früher als eine mit nur 12 Individuen, jedoch ist eine Größe von 24 Individuen sowohl schneller als eine mit 12, als auch eine mit 96 Individuen. Um eine effiziente Optimierung durchzuführen, sollte eine Populationsgröße von mindestens 24 Individuen gewählt werden und auf Grund der Untersuchung aus Kapitel 5.2 kleiner als die in diesem Fall zur Verfügung stehende Anzahl von 48 Kernen.

5.5 Optimierung eines Solarturm

Mit den ermittelten Werten kann nun eine Optimierung des bereits vorhandenen Solarturmkraftwerks PS10 simuliert werden, siehe Abbildungen 24 und 25.

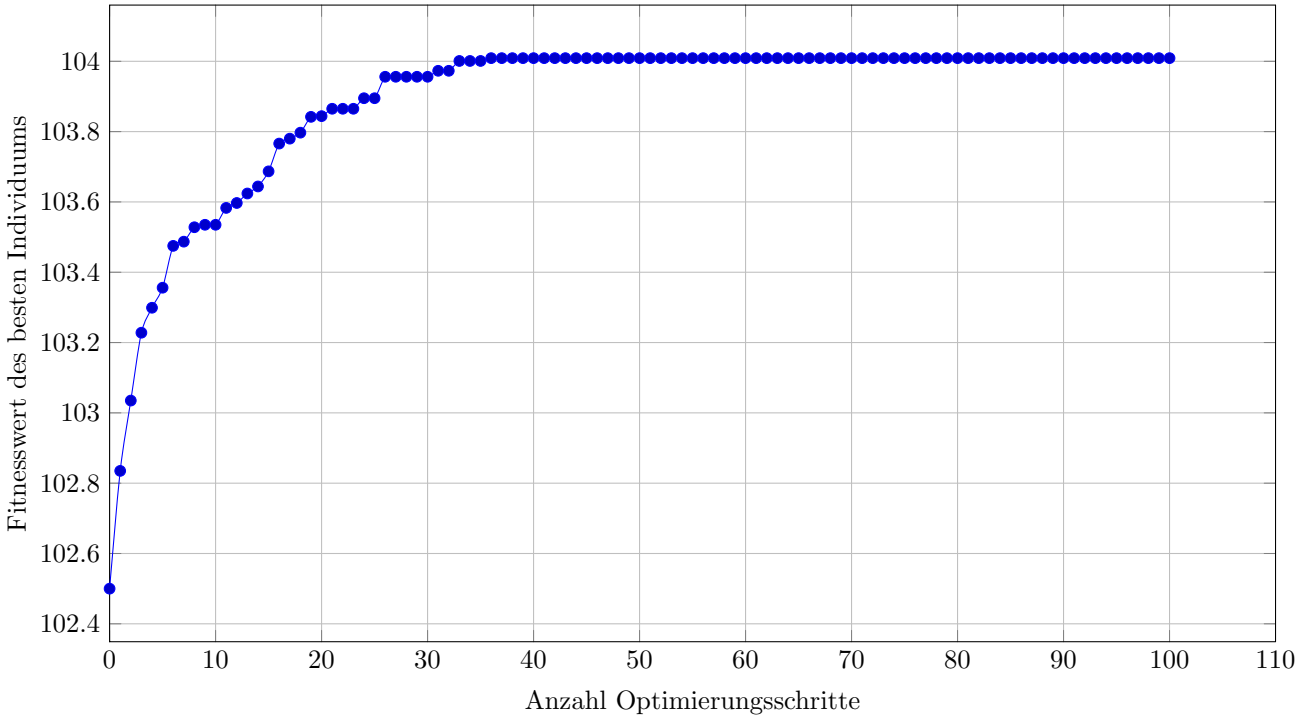
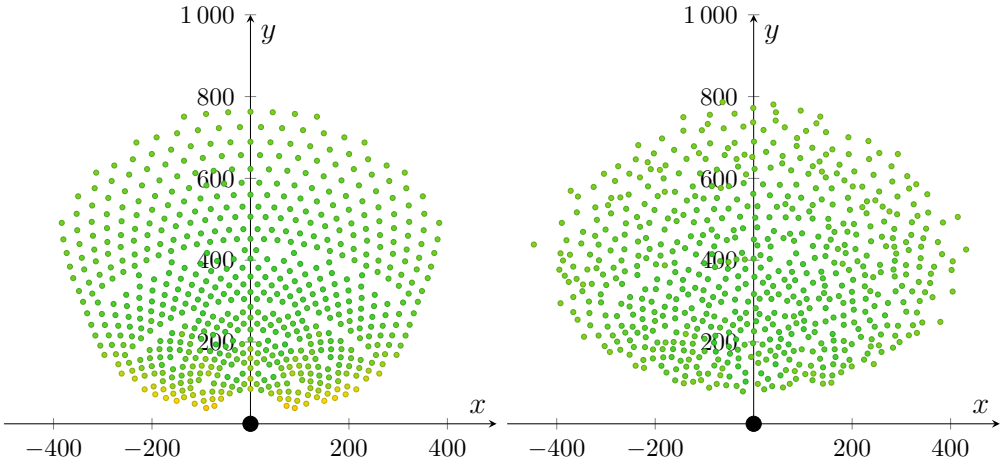
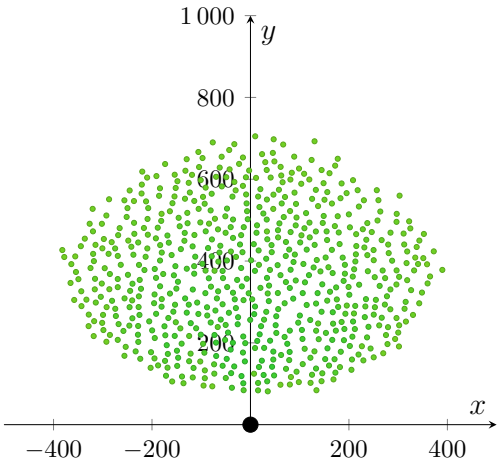


Abbildung 24: Optimierungsverlauf unter Verwendung des RayTrace Modells und eine Populationsgröße von 24 Individuen, einer Rastergröße von 15 m und 36 Jahrespunkten mit 7 Tagespunkten

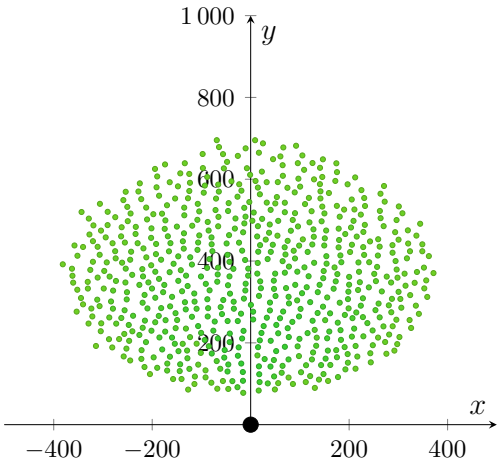


(a) PS10 Original Positionierung

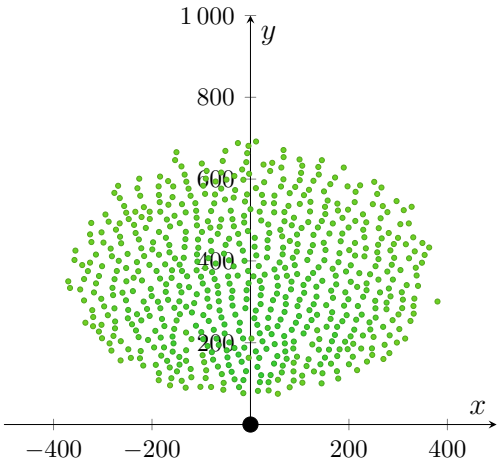
(b) Optimierungsschritt 2



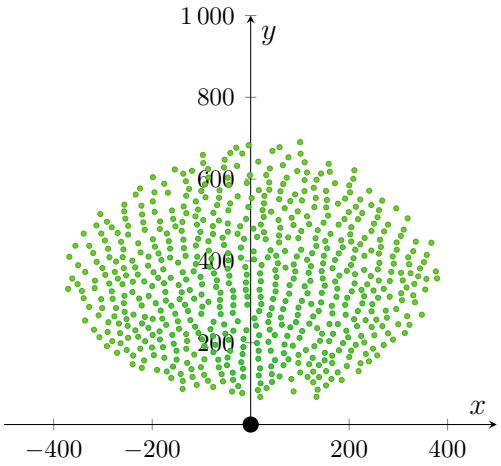
(c) Optimierungsschritt 10



(d) Optimierungsschritt 40



(e) Optimierungsschritt 80



(f) Optimierungsschritt 100

Abbildung 25: Visualisierung einzelner Optimierungsschritte

6 Zusammenfassung und Ausblick

Die Ergebnisse der Optimierung unter Verwendung der Testfunktionen hat gezeigt, dass die Optimierung nach dem Prinzip des Genetischen Algorithmus gute Werte liefern kann. Die Qualität der Ergebnisse bei der Optimierung eines realen Solarturms, wobei die Berechnung der Jahresleistung mit Hilfe eines Software Modells realisiert wird, ist dabei natürlich von der Genauigkeit dieses Modells abhängig.
Das hier verwendete RayTrace Modell wurde im Laufe der Arbeit an vielen Stellen überarbeitet und erweitert. Eine Verifikation des Modells fand jedoch noch nicht statt. Diese wäre für die Verwendung des Optimierungsalgorithmus notwendig, um einen erfolgreichen Einsatz zu gewährleisten. Zur Verifikation käme das Monte-Carlo Tool SolTRACE [9] in Frage.
Auch an dem Algorithmus sind noch Erweiterungen denkbar. Es werden hier z.B. nur die x- und y-Koordinaten betrachtet. Auf einem realen Gelände gibt es jedoch Höhenunterschiede und es müsste noch die z-Koordinate beachtet werden. Dazu wäre die Verwendung eines Höhenprofils notwendig.
Bei der Optimierung wird nur in Bezug auf einen Gentyp, die Position der Heliostaten, optimiert. Es gibt allerdings noch weitere Gene, welche in Betracht gezogen werden können, wie z.B. die Maße der Spiegel oder die Größe des Receivers.
Das Ziel dieser Arbeit war, die Energie, die ein Solarturm über einen Zeitraum von einem Jahr erbringen kann, zu maximieren. Die Minimierung der Stromerzeugungskosten wäre eine weiteres Modell, welches zur Optimierung verwendet werden könnte.

Literatur

[1] Udo E Simonis and Jörg Sommer. Das „doha climate gateway “. 2013.

[2] Bundesumweltministeriums BMU. Gemeinsame pressemitteilung des bundesumweltministeriums und des bundesministeriums für wirtschaftliche zusammenarbeit und entwicklung, nr. 084/12. 2012.

[3] Andrzej Graczyk. Nachhaltige entwicklung der energiewirtschaft im lichte der neuen eu- regelungen. *Nachhaltige Entwicklung–transnational. Sichten und Erfahrungen aus Mitteleuropa*, 16:249, 2011.

[4] Schaffen von Rahmenbedingungen für Innovation. 1.4 rahmen und struktur der energieforschung des bundes. *Forschung für eine umweltschonende, zuverlässige und bezahlbare Energieversorgung*, page 17.

[5] G. Morin. *Techno-economic design optimization of solar thermal power plants*. PhD thesis, Technische Universität Braunschweig, 2010.

[6] Wilhelm Hartmann and Richard Reinhard Emil Schorr. Beitrag zur geschichte und theorie der astronomischen instrumente mit rotierendem planspiegel und fester reflexrichtung:(heliostat, siderostat, zölostat, uranostat). *Astronomische Abhandlungen der Hamburger Sternwarte*, 4:1–36, 1928.

[7] Eberhard Schöneburg, Frank Heinzmann, Sven Feddersen, et al. *Genetische Algorithmen und Evolutionsstrategien*, volume 1. Addison-Wesley Reading, MA, 1994.

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

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

[10] Jean H Meeus. *Astronomical algorithms*. Willmann-Bell, Incorporated, 1991.

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

[12] M. Geyer and W.B. Stine. Power from the sun. Website, 2001. Available online at `www.powerfromthesun.net/book.html`, visited on October 1st 2012.

[13] Robert Pitz-Paal, Nicolas Bayer Botero, and Aldo Steinfeld. Heliostat field layout optimization for high-temperature solar thermochemical processing. *Solar Energy*, 85(2):334–343, 2011.

[14] PL Leary and JD Hankins. User's guide for mirval: a computer code for comparing designs of heliostat-receiver optics for central receiver solar power plants. Technical report, Sandia Labs., Livermore, CA (USA), 1979.

[15] Ari Rabl. *Active solar collectors and their applications.* Oxford University Press, USA, 1985.

[16] Rohit Chandra, Ramesh Menon, Leo Dagum, David Kohr, Dror Maydan, and Jeff McDonald. *Parallel programming in OpenMP.* Morgan Kaufmann, 2000.