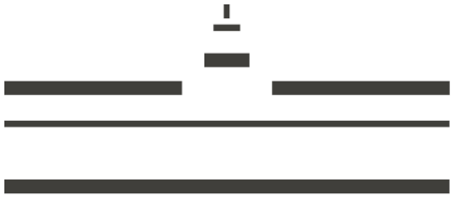


Westfälische
Wilhelms-Universität
Münster

Analyse intelligenter Haushaltsgeräte in Smart Homes zur Maximierung der Eigennutzung lokaler Stromproduktion

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science

Westfälische Wilhelms-Universität Münster
Fachbereich Mathematik und Informatik
Sicherheitskritische Systeme

Betreuung:
Prof. Dr. Anne Remke

Eingereicht von:
Anna-Lisa Linnemann

Münster, 20. November 2016

Inhaltsverzeichnis

1 Einleitung

Der Ausbau erneuerbarer Energien spielt in der Energiewende eine zentrale Rolle und hat in den letzten Jahren deutlich zugenommen. Lag der Anteil an erneuerbaren Energien in Deutschland 2014 noch bei zirka 26 %, so stieg dieser 2015 bereits auf rund 30 % [1] an. Der Ausbau hat auch in der Politik hohe Priorität. So sieht das aktuelle Energiekonzept eine Steigerung des Anteils der erneuerbarer Energien an der Stromerzeugung auf 40-45 % bis zum Jahr 2025 vor. Das Erneuerbare-Energien-Gesetz (EEG) unterstützt diesen Ausbau zwar, indem es die Förderung der erneuerbaren Energien durch den Einspeisevorrang und garantierte Einspeisevergütungen regelt, jedoch sinken diese Vergütungen immer weiter. Wind- und Sonnenenergie stellen den größten Anteil der erneuerbaren Energien dar. Letztere lässt sich in vieler Hinsicht direkt nutzen. So wird bei einer Photovoltaikanlage (PV-Anlage) mittels Solarzellen ein Teil der Sonnenstrahlung direkt in Strom umgewandelt [2], [1]. Für eine Privatperson ist dies eine gute Möglichkeit selbst Strom zu produzieren und zu nutzen.
Auch die Einspeisevergütung des von der PV-Anlage produzierten Stroms sinkt immer weiter [3, S. 11]. So lag diese für Neuanlagen unter 10 kWp Leistung (Ein- und Mehrfamilienhaus-Größe) 2012 noch bei ungefähr 19 Cent/kWh [4], 2015 jedoch nur noch bei zirka 12,5 Cent/kWh [5], wohingegen der Stromeinkaufspreis 2015 bei etwa 28,81 Cent/kWh [6] lag. Aufgrund dieser großen Diskrepanz zwischen der Einspeisevergütung und dem Einkaufspreis ist es von Vorteil den lokal produzierten Strom, wenn möglich, für den Eigenverbrauch zu nutzen. Das bedeutet, dass der lokal produzierte Strom möglichst zur Eigennutzung verwendet und nicht in das öffentliche Stromnetz eingespeist wird.

Aufgrund schwankender Sonneneinstrahlung im Verlauf des Tages kommt es zu einer ungleichmäßigen Stromproduktion der PV-Anlage. Somit kann es zu einem Stromüberschuss oder Strommangel kommen. Bei einem Stromüberschuss muss der überschüssige Strom in das öffentliche Stromnetz eingespeist werden. Daraus resultiert der Nachteil der oben genannte Diskrepanz zwischen Einspeisevergütung und Einkaufspreis als auch der Verlust von Strom durch die Übertragung. Es ist somit wünschenswert den Eigenverbrauch zu steigern.
Eine Steigerung lässt sich durch effiziente Energienutzung in sogenannten *Smart Homes* realisieren, d. h. in Wohnhäusern mit miteinander vernetzten elektronischen Verbrauchsgeräten [7]. Eine Besonderheit eines Smart Home ist, dass Verbrauchsmuster gewisser Haushaltsgeräte so ausgerichtet werden können, dass diese Muster an die Stromproduktion der PV-Anlage angeglichen werden. So kann der Waschvorgang einer

Waschmaschine beispielsweise auf eine Zeit verschoben werden, zu der ein Überschuss an lokal produziertem Storm vorhanden ist. Dazu kann ein Endzeitpunkt der Waschmaschine angegeben werden, sodass der Waschvorgang bis zu diesem Zeitpunkt beendet sein muss. Wann dieser Vorgang genau stattfindet, wird automatisch von einem Energiemanagementsystem (EnMS) kontrolliert [8]. Dieses System nutzt alle relevanten Informationen inklusive der Wettervorhersage, um den Verbrauch der steuerbaren Geräte anzupassen. Diese Geräte werden auch *smarte Geräte* genannt. Um den Eigenverbrauch des lokal erzeugten Stroms zu erhöhen, ist ein optimaler „Tagesablauf“, ein sogenannter optimaler *Schedule*, dieser smarten Geräte von Vorteil.
Eine andere Möglichkeit den Eigenverbrauch zu steigern, ist der Gebrauch einer Batterie als Energiespeicher. Diese kann den überschüssigen lokal produzierten Strom der PV-Anlage speichern und diesen zu Zeiten geringer Stromproduktion wieder bereitstellen. Ein Nachteil hierbei ist der Lade- bzw. Entladeverlust der Batterie. Eine Kombination beider Möglichkeiten scheint am effizientesten, um den Eigenverbrauch zu maximieren.

Um die Auswirkungen dieser Möglichkeiten auf den Eigenverbrauch zu untersuchen, wird in dieser Arbeit unter anderem der optimale Schedule der smarten Geräte berechnet. Dieser Schedule ist kein realistischer, sondern ein idealer. D. h. alle relevanten Informationen, wie beispielsweise die Stromproduktion, sind im Voraus bekannt. Für diese Berechnung wird das Tool UPPAAL CORA [9] verwendet, welches zur optimalen Kostenerreichbarkeitsanalyse genutzt wird. In diesem Tool wird ein sogenanntes *smartes Modell* erstellt, in dem die smarten Geräte als ein Netzwerk aus *Linearly Priced Timed Automata* (LPTA) modelliert werden. Die Auswertung dieses Modells liefert also einen optimalen Schedule. Neben diesem Modell wird in dieser Arbeit ein sogenanntes *konventionelles Vergleichsmodell* betrachtet, welches keine smarten Geräte besitzt. Um die Auswirkungen einer Batterie auf den Eigenverbrauch beurteilen zu können, werden zudem diese Modelle zusätzlich mit einer Batterie betrachtet, sodass vier Modelle untersucht werden.

Als smarte Geräte werden unter anderem eine Waschmaschine und ein Wäschetrockner betrachtet. Bei diesen kann die Zeit zwischen dem Auftrags- und Endzeitpunkt stark variieren. Dies kann Einfluss auf den Eigenverbrauch haben. So ist es zum Beispiel bei einer Zeitspanne von sechs Stunden wahrscheinlicher, dass der Waschvorgang zu einem Zeitpunkt lokal produziertem Stromüberschuss stattfindet, als bei einer Zeitspanne von nur drei Stunden. Um die Folgen dieser unterschiedlichen Intervalllängen beurteilen zu können, werden in den smarten Modellen verschiedene feste Intervalllängen untersucht.

Damit sich die verschiedenen Modelle vergleichen lassen, wird für jedes Modell der Eigenverbrauch, die Stromkosten und die Deckung des Stromverbrauchs der betrachteten Geräte durch den lokal erzeugten Strom berechnet. Für aussagekräftige Resultate werden zudem verschiedene Haushalte über ein ganzes Jahr betrachtet. Die benötigten Produktions- und Verbrauchsdaten werden dafür von dem künstlichen Lastprofilgenerator ALPG (engl. *artifical load profile generator*) [10] geliefert.

In der Bachelorarbeit „Synthesis of smart appliances in smart homes with batteries to maximize self-use“ von Fabian Stein [11] wurde ein ähnliches Modell untersucht. In dieser Arbeit wurde ebenfalls ein Smart Home mit PV-Anlage und ähnlichen smarten Geräten betrachtet, die als ein Netzwerk aus LPTAs in UPPAAL CORA modelliert wurden. Es wurde der Einfluss von einer Batterie und dem Schedule smarter Geräte in einem privaten Haushalt im Hinblick auf die Maximierung des Eigenverbrauchs analysiert. Dazu wurden verschiedene Batteriegrößen zwischen 0,25 kWh und 7 kWh untersucht. Aufgrund des ähnlichen Modells wird im Folgenden immer wieder auf diese Arbeit eingegangen, um die Unterschiede darzustellen.

Diese Arbeit ist wie folgt strukturiert: In Kapitel 2 wird auf die theoretischen Grundlagen eingegangen. Es wird der Lastprofilgenerator ALPG vorgestellt, sowie das verwendete Tool UPPAAL CORA. Die Modellannahmen der Haushaltsprofile, der smarten Geräte und der Batterie sind Thema von Kapitel 3. Von der Implementierung und Modellierung wird in Kapitel 4 berichtet. Es werden die Datengewinnung und -extrahierung aus dem Tool ALPG für die einzelnen Haushalte beschrieben, sowie die Modellierung des smarten Modells in UPPAAL CORA und die Berechnung des optimalen Schedules. Zudem wird auf die Implementierung des konventionellen Modells, der Batterie und die Analyse eingegangen. Daraufhin werden in Kapitel 5 die Ergebnisse dieser Analyse vorgestellt und untersucht. Abschließend werden in Kapitel 6 die relevanten Ergebnisse zusammengefasst.

2 Theoretische Grundlagen

2.1 ALPG

ALPG ist ein künstlicher Lastprofilgenerator (engl. *artifical load profile generator*), der von Hoogsteen et al. [12] in Python entwickelt wurde. Er synthetisiert Energieprofile für verschiedene Haushaltstypen für die Analyse des Lastmanagements, dem sogenannten *Demand Side Management* (DSM) [13, S.4]. DSM ist die „Einflussnahme durch den Energieversorger bzw. Dritte auf die Energienachfrage (i.d.R. Stromnachfrage, d.h. die Last) von Konsumenten (z.B. Haushalten oder Industrie) zur Steuerung der Energiemenge oder den Zeitpunkt des Energiekonsums“ [14].

ALPG nutzt ein Bottom-up Konzept. Dieses legt zunächst für jeden Haushalt die verfügbaren flexiblen bzw. smarten Geräte, die vorhandenen Installationen erneuerbarer Energien, die verfügbaren Batterien und die Konfiguration der Bewohner fest. Zu den smarten Geräten können beispielsweise eine Waschmaschine, eine Spülmaschine oder ein Elektroauto zählen und zu den erneuerbaren Energien eine Photovoltaikanlage.
Das Verhalten der einzelnen Bewohner wird simuliert, um ein Nutzungsprofil zu erstellen, welches wiederum genutzt wird, um konsistente Profile für die Geräte zu erzeugen. Auf diese Weise kann der Zustand eines Gerätes, welches eine Eingabe eines Benutzers benötigt, nur geändert werden, wenn eine Person zu Hause ist. Auch die Start- und Endzeiten der flexiblen Geräte werden mit dem Nutzungsprofil synchronisiert.
Die statischen, unflexiblen Geräte werden in die folgenden Kategorien unterteilt: Elektronik, Kühlschrank, Induktiv, Beleuchtung, Andere und Standby [12, S.2-3].

Es können mehrere Parameter für das Modell konfiguriert werden. Dadurch kann zwischen möglichen erneuerbaren Energien und unterschiedlichen Haushaltstypen in der Nachbarschaft variiert werden. Folgende Parameter müssen angegeben werden:

- Simulationsparameter: Die Anzahl der zu simulierenden Tage.
- Erneuerbare Energien: Jeweils der prozentuale Anteil an Haushalten, die ein Elektroauto (FEV), einen Plug-in-Hybrid (PHEV), einen Batteriespeicher oder eine Photovoltaik-Dachanlage (PV-Anlage) besitzen.
- Stromverbrauch verschiedener Geräte: Leistungen und, wenn vorhanden, Batteriekapazitäten der erneuerbaren Energien; Verbrauchslevel anspruchsvoller Geräte, wie beispielsweise einem Induktionsherd.

- Geographische Lage der Nachbarschaft: Die geographische Lage, um Sonnenauf- und -untergang zu erhalten.
- Eingabedatei der Sonneneinstrahlung: Die Daten der Sonneneinstrahlung, welche für die Berechnung der Stromproduktion der PV-Anlage benötigt wird.
- Haushaltstypen in der Nachbarschaft: Die Anzahl und Art der Haushalte wird durch die Anzahl der vordefinierten Haushaltstypen angegeben (siehe Tabelle 1).
- Vorhersagbarkeit der Personen: Die Vorhersagbarkeit der Bewohner kann konfiguriert werden, indem Zufallsereignisse zufällig zugelassen werden.

In Tabelle 1 sind die verschiedenen Haushaltstypen dargestellt. In der ersten Spalte ist der Name des Typs aufgeführt und in der zweiten Spalte die dazugehörige Beschreibung. Desweiteren ist in der Tabelle der Jahresverbrauch für jeden Haushaltstyp in kWh und die Anzahl der Personen, die in dem Haushalt leben, dargestellt. Die Zahl in den Klammern gibt dabei an, wie viele der Personen erwachen sind.

Name	**Beschreibung**	**Jahresverbrauch** (in kWh)	**Personen (Erwachsene)**
SingleWorker	alleinlebend, berufstätig	1610 - 2410	1 (1)
DualWorker	zwei berufstätige Personen	2660 - 4060	2 (2)
FamilyDualWorker	Familie, berufstätige Eltern	3460 - 7060	3 - 6 (2)
FamilySingleWorker	Familie, ein berufstätiges Elternteil	3460 - 7060	3 - 6 (2)
FamilySingleParent	Familie, alleinerziehend	2600 - 6200	2 - 5 (1)
DualRetired	Rentnerpaar	2660 - 4060	2 (2)
SingleRetired	alleinlebender Rentner	1610 - 2410	1 (1)

Tabelle 1: Vordefinierte Haushaltskonfigurationen.

Es gibt zwei verschiedene Ausgabearten: den flexiblen und unflexiblen Teil. Zum Letzteren zählen beispielsweise der Stromverbrauch und die Stromproduktion der möglichen PV-Anlage. Er beinhaltet zum einen den durchschnittlichen Strombedarf in Watt für jede Minute und jeden Haushalt, zum anderen die Blindleistung in Var für jede Minuten und jeden Haushalt. Beim flexiblen Teil werden beispielsweise die Start- und Endzeiten der flexiblen Geräte, wie der Wasch- oder Spülmaschine, in Sekunden angegeben. Das Gerät sollte folglich nicht vor der Startzeit beginnen und vor der Endzeit fertig sein. [12, 4-5]

2.2 UPPAAL CORA

UPPAAL ist ein Werkzeug für die Modellierung, Simulation und Verifikation von Echtzeitsystemen und wird an der Universität Uppsala und der Universität Aalborg entwickelt [15, S.1].
UPPAAL CORA ist eine Erweiterung dieses Werkzeugs zur optimalen Kostenerreichbarkeitsanalyse. Sie wurde vom UPPAAL-Team als Teil der Projekte VHS (Verification of Hybrid Systems) und AMETIST (Advanced Method for Timed Systems) entwickelt. Während UPPAAL Echtzeitautomaten als Eingabe analysiert, arbeitet UPPAAL CORA basierend auf dem Formalismus der Linearly Priced Timed Automata (LPTA) bzw. dem der Netzwerke von LPTAs (NLPTAs). Letztere bestehen aus mehreren LPTAs, die parallel laufen und miteinander kommunizieren. Durch die zusätzliche Angabe der Kosten für Transitionen und Zustände, findet UPPAAL CORA den optimalen, kostenminimierten Pfad unter Berücksichtigung von festgelegten Zielbedingungen [16].

2.2.1 Aufbau

UPPAAL CORA besteht aus drei Komponenten, welche die Aufgaben des Modellierens im Editor, der Simulation und der Verifikation übernehmen.
Der Editor ist in Abbildung 1 dargestellt und liefert eine graphische Oberfläche, um die LPTAs bzw. NLPTAs zu modellieren.

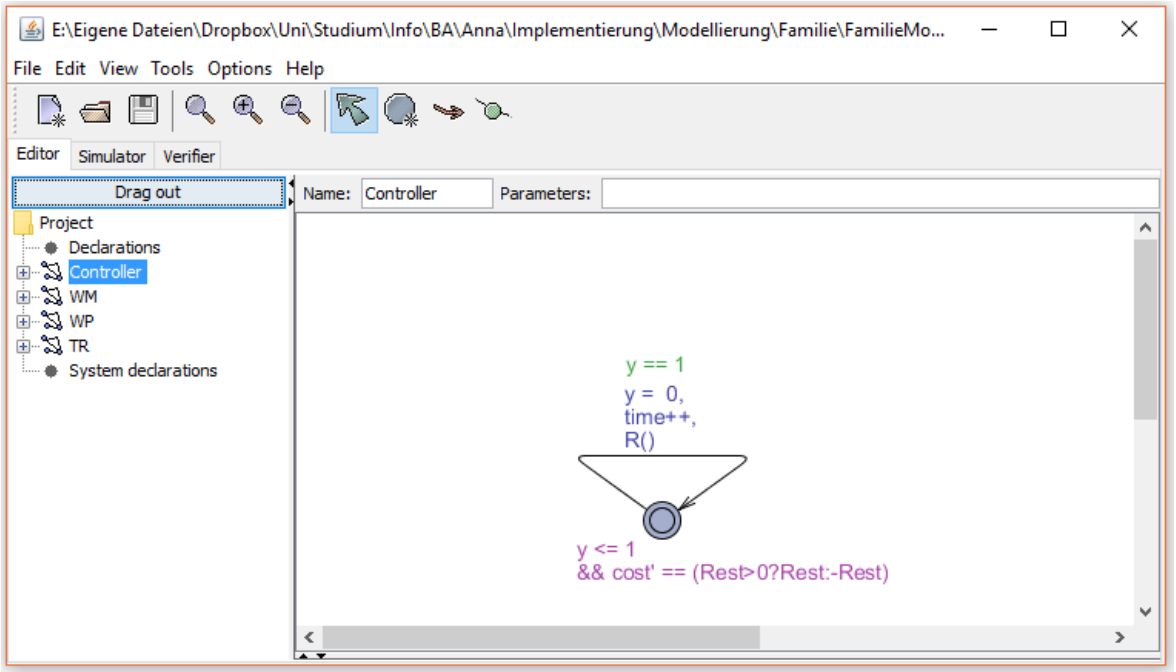


Abbildung 1: Editor von UPPAAL CORA.

Zudem ist es möglich, das Systemverhalten dieser Netzwerke mithilfe der Beschreibungssprache und Datenvariablen darzustellen. Die verwendete Beschreibungssprache ist eine nicht-deterministische *guarded command* Sprache mit Datentypen [15, S.1].
Im Simulator werden die aktuellen Zustände und die möglichen Übergänge in den verschiedenen LPTAs dargestellt, siehe Abbildung 2. Es ist somit möglich, sich schrittweise durch das System zu bewegen. Dabei werden die Veränderungen der Variablen und Kosten in dem Fenster in der Mitte angezeigt.

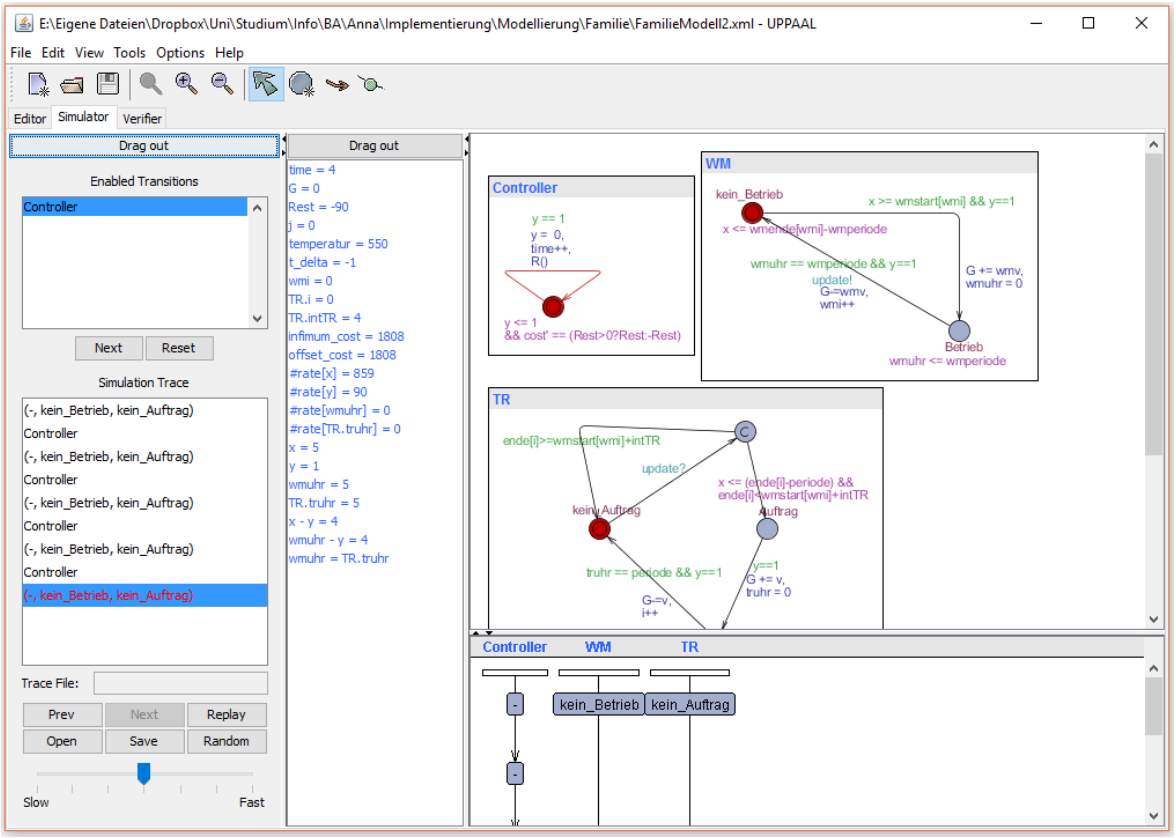


Abbildung 2: Simulator von UPPAAL CORA.

Die Verifikation findet im Model-Checker statt. Dieser ist in Abbildung 3 dargestellt. Dem Model-Checker können Anfragen in temporaler Logik in Form von sogenannten *Queries* gestellt werden, die anschließend gelöst werden. In Tabelle 2 ist eine Übersicht der häufigsten Queries gegeben [17, S.2-9].
Wird die Option `Diagnostic Trace` ausgewählt, so kann bei der gelungenen Untersuchung einer Pfadgenerierung, dieser Pfad im Simulator dargestellt werden. Es gibt die Option einen beliebigen Pfad (engl. *some Trace*) zu berechnen oder den *besten* Pfad (engl. *best Trace*), d. h. den kostenoptimierten Pfad. Beispielsweise wird bei den Queries der Form `E<> p` ein Pfad erstellt, wenn die Eigenschaft `p` erfüllt ist [18].

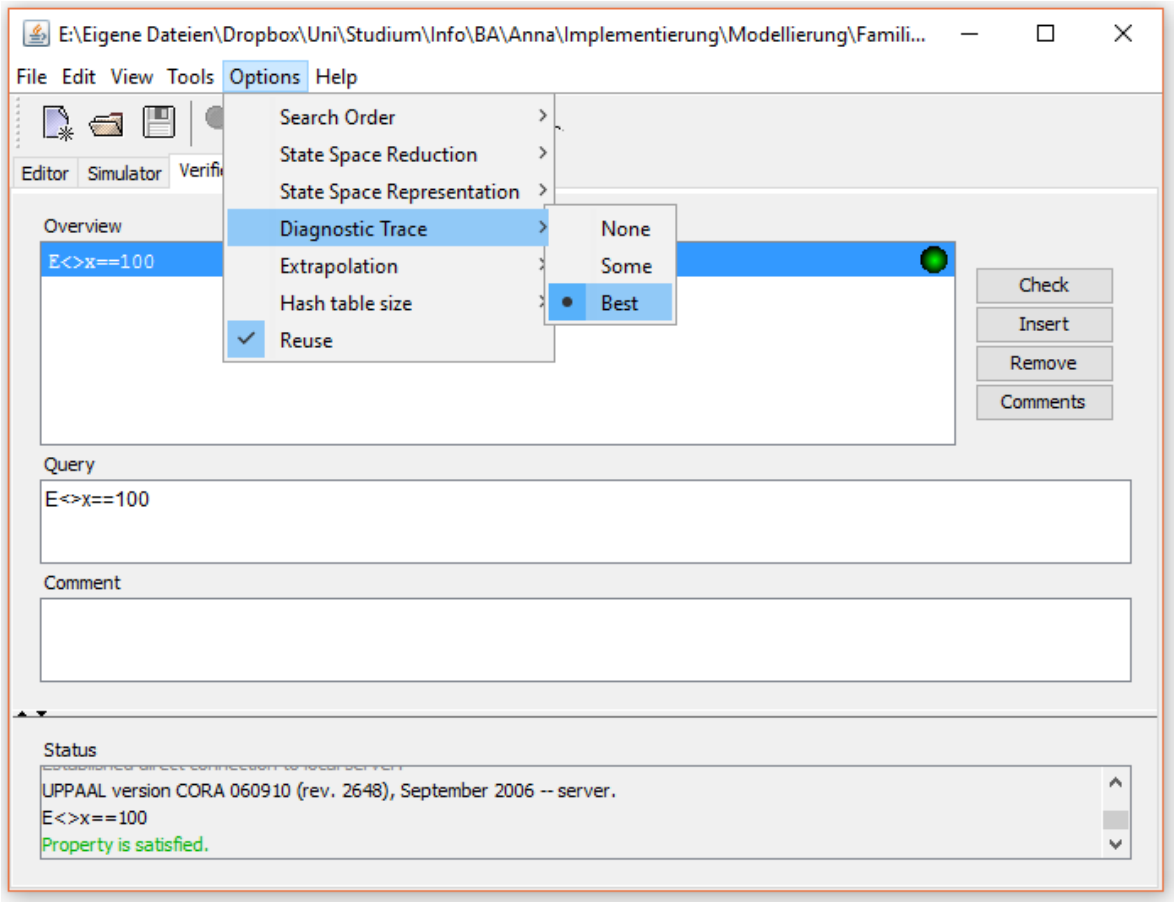


Abbildung 3: Model-Checker von UPPAAL CORA.

E<> p	es existiert ein Pfad, in dem p erfüllt sein kann
A[ ] p	für alle Pfade ist p immer erfüllt
E[ ] p	es existiert ein Pfad, in dem p immer erfüllt ist
A<> p	für alle Pfade kann p erfüllt sein
p - -> q	immer wenn p erfüllt ist, kann q auch erfüllt sein

Tabelle 2: Häufige Queries des Model-Checkers.

Das Berechnen großer Verifikationsaufgaben innerhalb der GUI von UPPAAL CORA ist häufig unzweckmäßig. Für solche Situationen gibt es den Kommandozeilen-Verifizierer *verifyta*. Dieser akzeptiert Kommandozeilen-Argumente für alle Optionen, die auch in der GUI vorhanden sind. Die wichtigsten Optionen in Bezug auf UPPAAL CORA sind in Tabelle 3 dargestellt. Die Bestensuche *Best first* und die beste Tiefensuche *Best depth first* werden für die optimale Erreichbarkeit genutzt. Auf die Bedeutung der Suchstrategie wird in Kapitel 2.2.3 näher eingegangen.
Der optimale Pfad kann mit der Option `-t3` gefunden werden. UPPAAL CORA nutzt

automatisch die Bestensuche (`-o4`), wenn die Option `-t3` genutzt wird. Dies kann durch die Angabe der verschiedenen `-o`-Optionen nach der `-t3`-Option geändert werden [19]. Zusätzlich zu den Optionen muss die Datei des UPPAAL-CORA-Modells und die der gewünschten Query angegeben werden.

Suchstrategien	
-o0	Breadth first
-o1	Depth first
-o2	Random depth first
-o3	Smallest heur first
-o4	Best first
-o5	Best depth first
Pfadoptionen	
-t0	Some Trace
-t3	Best Trace
-E	findet die optimalen Kosten ohne Generierung eines Pfades

Tabelle 3: Optionen des Verifyta und die dazugehörigen Optionen in der GUI.

2.2.2 Linearly Priced Timed Automata

Ein Linearly Priced Timed Automaton (LPTA) ist eine Erweiterung der Echtzeitautomaten (engl. *timed automata*) mit zusätzlichen Kosteninformationen für die Zustände und Transitionen. Er ist daher sehr nützlich, um Scheduling-Probleme für Echtzeitsysteme zu modellieren. Die Kostenvariable einer Transition gibt an, um wie viel die Kosten steigen, wenn diese ausgeführt wird. Die Kostenvariable eines Zustandes stellt die Rate pro Zeiteinheit dar, um in dem Zustand zu bleiben. Ist die Rate in einem Zustand beispielsweise 3 und es sind 1,3 Zeiteinheiten in diesem Zustand vergangen, so erhöhen sich die Kosten um das Produkt 3,9 [20, S.3-4].

2.2.2.1 Syntax und Semantik

C sei eine endliche Menge aus Uhren und $\mathcal{B}(C)$ die Menge aller Formeln, die aus Konjunktionen von atomaren Bedingungen der Form $x \bowtie n$ mit $x \in C$, $n \in \mathbb{N}$ und $\bowtie \in \{<, \leq, =, \geq, >\}$ bestehen. Die Elemente dieser Menge werden Uhrenbedingungen über C genannt [20, S.4].

Definition 2.1 *(Linearly Priced Timed Automaton)*
Ein LPTA über eine Uhrenmenge C und einer endlichen Eingabemenge Act ist ein Tupel (L, l_0, E, I, P), wobei gilt:

- *L ist eine endliche Menge von Zuständen,*

- l_0 *ist der Anfangszustand,*
- $E \subseteq L \times \mathcal{B}(C) \times Act \times \mathcal{P}(C) \times L$ *ist eine Zustandsübergangsrelation,*
- $I : L \to \mathcal{B}(C)$ *ordnet jedem Zustand eine Invariante zu,*
- $P : (L \cup E) \to \mathbb{N}$ *ordnet Zuständen und Zustandsübergangsrelationen Kosten zu.*

Ein Element $e = (l_1, g, a, r, l_2) \in E$ bezeichnet eine Transition von l_1 nach l_2 mit einer einschränkenden Uhrenbedingung g, einer Eingabe a und einer Menge r von Uhren, die zurückgesetzt werden. Eine weitere Schreibweise ist $l_1 \xrightarrow{g,a,r} l_2$.
Die Uhrenwerte werden als Funktionen von C in die nicht-negativen reellen Zahlen aufgefasst und Uhrenbelegungen genannt, wobei mit $\mathbb{R}^C$ $(u_1, u_2 \in \mathbb{R}^C)$ die Menge aller Uhrenbelegungen für C bezeichnet wird [20, S.4-5].

Definition 2.2 *(Semantik eines LPTA)*
Die Semantik eines LPTA wird als ein Transitionssystem im Zustandsraum $L \times \mathbb{R}^C$ *mit Anfangszustand* (l_0, u_0) *(*u_0 *weist allen Uhren in* C *den Wert 0 zu) und folgender Transitionsrelation definiert:*

- $(l_1, u_1) \xrightarrow{\epsilon(d),p} (l_1, u_1 + d)$, *wenn* $u_1 + d \in I(l_1)$ *und* $p = P(l_1) * d$,
- $(l_1, u_1) \xrightarrow{a,p} (l_2, u_2)$, *wenn* $\exists\ g, r$, *sodass* $l_1 \xrightarrow{g,a,r} l_2$, $u_1 \in g$, $u_2 = [r \mapsto 0]u_1$, $u_2 \in I(l_2)$ *und* $p = P((l_1, g, a, r, l_2))$.

Durch die Operation $[r \mapsto 0]u_1$ wird jede Uhr $x \in r$ auf den Wert 0 gesetzt und alle anderen Uhren $x \in \mathcal{P}(C) \backslash r$ auf den Wert $u_1(x)$. Zudem weist die Operation $u_1 + d$ mit $d \in \mathbb{R}_{\geq 0}$ jeder Uhr $x \in \mathcal{P}(C)$ den Wert $u_1(x) + d$ zu.
Der erste Transitionstyp der Relation, die sogenannte *delay transition*, entspricht der Verweildauer in einem Zustand. Dieser bleibt somit gleich und die Werte der Uhren werden uniform um d erhöht, wobei die neuen Werte der Uhren die Invariante des Zustands weiterhin erfüllen müssen.
Der zweite Transitionstyp, die sogenannte *action transition*, enspricht der Ausführung einer Transition aus E, wobei die Invariante des neuen Zustands weiterhin erfüllt sein muss [21, S.5].
Die Kosten eines Ablaufs der Transitionsrelation sind die addierten Kosten der besuchten Zustände und ausgeführten Transitionen.

2.2.2.2 Kostenoptimales Erreichbarkeitsproblem

Das Problem der kostenoptimalen Erreichbarkeit (engl. *cost-optimal reachability*) ist es, das Minimum der notwendigen Kosten zu finden, um einen gegeben Zielzustand zu erreichen.

Definition 2.3 *(Kosten)*
Sei $\alpha = (l_0, u_0) \xrightarrow{a_1,p_1} (l_1, u_1) \cdots \xrightarrow{a_n,p_n} (l_n, u_n)$ *eine endliche Ausführung eines LPTA. Dann stellt* $cost(\alpha)$ *die Kosten von* α *dar, welche durch die Summe* $\sum_{i\in\{1,\cdots n\}} p_i$ *berechnet werden.*
Für einen gegebenen Zustand (l, u) *ist* $mincost(l, u)$ *das Minimum der Kosten, um* (l, u) *zu erreichen und ist somit das Infimum der Kosten aller endlichen Abläufe, die in* (l, u) *enden. Das Minimum der Kosten, die notwendig sind, um den Zustand* l *zu erreichen, ist das Infimum der Kosten aller endlichen Ausführungen, die in einem Zustand der Form* (l, u) *enden, und wird mit* $mincost(l)$ *bezeichnet [20, S.5], d.h.*

$$mincost(l) := inf\{cost(\alpha) | \alpha \text{ ist eine Ausführung, die im Zustand } l \text{ endet}\}.$$

Das kostenoptimale Erreichbarkeitsproblem eines LPTA mit gegebenem Zustand l ist es somit, die größten Kosten k zu finden, sodass $k \leq mincost((l, u))$ für alle Uhrenwerte u gilt.

Die Definition der Uhren über die nichtnegativen reellen Zahlen kann bei einem LPTA zu einem überabzählbaren Transitionssystem führen, sodass eine Suche in diesem ungeeignet ist. Daher werden die sogenannten *symbolischen Zustände* eingeführt, die symbolische Darstellungen möglicher infiniter Mengen aktueller Zustände und ihrer Verbindung zu den Kosten liefern. Dahinter steckt der Gedanke, dass während der Erkundung das Infimum der Kosten eines symbolischen Pfads (Pfad symbolischer Zustände) in dem symbolischen Zustand selbst gespeichert wird. Wenn der gleiche Zustand mit abweichenden Kosten entlang eines anderen Pfads erreicht wird, sollen die symbolischen Zustände verglichen und der Zustand mit den höheren Kosten verworfen werden. Formal besteht ein symbolischer Zustand eines LPTA aus einem Zustand und einer sogenannten *Priced Zone*.

Definition 2.4 *(Priced Zone)*
Eine Priced Zone über eine Uhrenmenge C *ist ein Tupel* (Z, f)*, wobei gilt:*

- Z *ist eine sogenannte Zone, das heißt eine Konjunktion der Uhrenbedingungen oder -differenzen,*

- *f ist eine affine Funktion über C, die die Kosten der Uhrenbelegungen berechnet, die die Bedingungen von Z erfüllen.*

Zu jeder Uhrenbelegung in der Zone werden somit die Kosten zu dieser Belegung geliefert.
Zonen können mithilfe von *Difference Bound Matrices* dargestellt und effizient manipuliert werden. Das genaue Vorgehen wird hier nicht weiter betrachtet und von David L. Dill in [22] formuliert.
Hiermit kann unter anderem der Nachfolger eines symbolischen Zustands, sowie die Dominanz zweier gegebener symbolischer Zustände, berechnet werden. Ein symbolischer Zustand S wird von S' dominiert, wenn alle Zustände $s \in S$ auch in S' sind und die Kosten von s in S' kleiner oder gleich der Kosten in S sind [23, S.6-7].

2.2.3 Branch-and-Bound-Algorithmus

UPPAAL CORA löst das Problem der kostenoptimalen Erreichbarkeit mithilfe des Branch-and-Bound-Algorithmus über die Menge der symbolischen Zustände der LPTAs. Das Verzweigen (engl. *branching*) basiert auf verschiedenen Suchstrategien, die in UPPAAL CORA implementiert sind. Dazu gehören z. B. die Breitensuche (engl. *beadth-first*), Tiefensuche (engl. *depth-first*), Bestensuche (engl. *best-first*) oder beste Tiefensuche (engl. *best depth-first*) [23, S.8-9]. Bei der Bestensuche wird der Zustand mit den kleinsten Werten für die Kostenvariable als nächstes gesucht. Wohingegen die beste Tiefensuche eine Tiefensuche ist, die immer den Zustand mit den niedrigsten Kosten zuerst sucht. [19]
Die Menge der möglichen symbolischen Zustände wird durch den Branch-and-Bound-Algorithmus in Algorithmus 1 überprüft und erweitert, bis der Zustand gefunden wird, der den kostengünstigsten expliziten Zielzustand darstellt.

Es gibt eine *Abgearbeitet*-Liste symbolischer Zustände, die bereits untersucht wurden. Diese Liste wird in Zeile 2 als leere Menge erzeugt. Weiterhin existiert eine *Wartend*-Liste symbolischer Zustände, die noch untersucht werden müssen. Diese wird in Zeile 3 mit dem symbolischen Anfangszustand S_0 initialisiert. Die Variable *Kosten* enthält die bisher geringsten Kosten aller betrachteten Pfade und wird anfänglich in Zeile 1 auf unendlich gesetzt.
Der Algorithmus iteriert, wie in Zeile 4 zu sehen, über die Menge der noch nicht überprüften symbolischen Zustände. In Zeile 5 wird in Abhängigkeit der gewählten Branching-Strategie ein symbolischer Zustand S aus der *Wartend*-Menge ausgewählt. Zudem wird überprüft, ob S weiter betrachtet werden muss oder verworfen werden

Algorithmus 1 Branch-and-Bound-Algorithmus

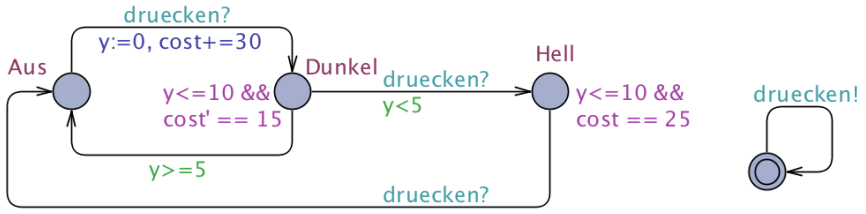
```
1:  Kosten ← ∞                          # enthält die bisher geringsten Kosten aller betrachteten Pfade
2:  Abgearbeitet ← ∅                    # schon betrachtete symbolische Zustände
3:  Wartend ← {S_0}                     # noch zu überprüfende symbolische Zustände
4:  while Wartend ≠ ∅ do                                          # Branching-Strategie
5:      wähle S ∈ Wartend
        C ← inf(S)
6:      if Abgearbeitet dom. nicht S then
7:          Abgearbeitet ← Abgearbeitet ∪ {S}
8:      else if S ∈ Ziel ∧ C < Kosten then
9:          Kosten ← C
10:     else
11:         Wartend ← {S'|S' ∈ Wartend ∨ S → S'}
12:     end if
13: end while
14: return Kosten
```

kann. Dabei wird in Zeile 6 S verworfen, wenn der Zustand von einem symbolischen Zustand aus der *Abgearbeitet*-Menge dominiert wird. Ansonsten wird S in Zeile 8 zu dieser hinzugefügt.
Wenn S ein Zielzustand mit geringeren als bisher gefundenen Kosten ist (siehe Zeile 8), werden die Kosten in Zeile 9 aktualisiert. Andernfalls werden die Nachfolgezustände von S in Zeile 11 der *Wartend*-Menge hinzugefügt und mit der Iteration fortgefahren. Der optimale Pfad wird nach Ausführung des Algorithmus rückwirkend rekonstruiert.

2.2.4 Linearly Priced Timed Automata in UPPAAL CORA

Im Folgenden wird das Modell der LPTA bzw. NLPTA in UPPAAL CORA erläutert und zum Teil anhand von Abbildung 4 verdeutlicht [24, S.4].

Diese Abbildung zeigt eine einfache Lampe, die mithilfe eines Schalters (Abbildung 4a) durch einen Akteur (Abbildung 4b) betätigt werden kann. Die Lampe leuchtet entweder hell, leicht oder ist aus, was durch die *Zustände* `Hell`, `Dunkel` und `Aus` repräsentiert wird. Da die Lampe zum Anfangszeitpunkt ausgeschaltet ist, ist der Startzustand `Aus`. Dieser wird in UPPAAL mit *Initial* bezeichnet und ist an dem doppelten Kreis zu erkennen.
In UPPAAL gibt es zudem zwei weitere Zustände: zum einen den sogenannten *urgent* Zustand (Zuordnung `u`) und zum anderen den sogenannten *committed* Zustand (Zuord-



(a) Lampe (b) Akteur

Abbildung 4: Modell einer Lampe (Abbildung 4a) mit einem Akteur (Abbildung 4b), der den Lichtschalter der Lampe betätigen kann

nung `c`). In beiden darf keine Zeit vergehen, wenn ein Prozess in dem jeweiligen Zustand ist. Der Unterschied der beiden Zustände liegt darin, dass der *committed* Zustand bei der nächsten Transition direkt verlassen werden muss. Somit ist dieser restriktiver als der *urgent* Zustand [25, S.6].

Weiterhin kann an einem Zustand eine Invariante angegeben werden, wie zum Beispiel $\texttt{y} \leq \texttt{10}$ an den Zuständen `Hell` und `Dunkel` (siehe Abbildung 4).
Eine Invariante ist nebenwirkungsfrei und kann nur aus Uhren, Integer-Variablen, Konstanten und Operatoren bestehen. Ein Prozess kann sich nur in einem Zustand befinden, wenn die zugehörige Invariante erfüllt ist [25, S.7].

Die Zustandsübergangsrelationen werden in UPPAAL als Kanten dargestellt.
Diese besitzen sogenannte *Select-Label*. Durch diese ist es möglich, einem Wert nichtdeterministisch einen Bezeichner aus dem jeweiligen Wertebereich zuzuweisen. Dieser kann daraufhin in den anderen Labels verwendet werden [25, S.6].

Die Kanten können zudem einschränkende Bedingungen, die sogenannten *Guard-Labels*, besitzen. Anders als bei der formalen Definition eines LPTA können diese nicht nur aus Uhrenbedingungen bestehen, sondern auch aus Datenbedingungen. Diese sind von ähnlicher Form, nutzen jedoch Integer-Variablen. Eine Kante kann nur genommen werden, wenn der Guard erfüllt ist [15, S.3]. Die Kante $\texttt{Dunkel} \rightarrow \texttt{Aus}$ ist beispielsweise mit dem Guard $y \geq 5$ ausgestattet, sodass die Lampe erst ausgeschaltet werden kann, wenn mindestens 5 Zeiteinheiten seit der ersten Schalterbetätigung vergangen sind.

Weiterhin können Kanten *Synchronisationslabels* besitzen, damit LPTAs untereinander synchronisiert werden können. Dies geschieht über sogenannte *Channels*. Zwei LPTAs können sich dabei über freigegebene Kanten synchronisieren, d. h. über Kanten, die

Guards erfüllen, und komplementäre Synchronisationslabels besitzen. Eine Kante besitzt einen Sender (markiert durch `!`) auf einem Channel und die andere Kante den Empfänger (markiert durch `?`) auf dem gleichen Channel. Bei einer Synchronisation werden beide Kanten zur selben Zeit genommen. Das bedeutet, die derzeitigen Zustände beider Automaten ändern sich.
Auch in Abbildung 4 gibt es eine Synchronisation zwischen den beiden Automaten. Die Kante des Akteur-LPTAs besitzt den Sender auf dem Kanal `druecken`, wohingegen die Kante `Hell` → `Aus` des Lampen-Automaten den korresponierenden Empfänger besitzt. Beide Kanten können somit nur synchron genommen werden.
Zudem gibt es zwei weitere Channel-Arten:
Durch die sogenannten *broadcast Channels* ist es möglich, dass mehrere Kanten in verschiedenen LPTAs miteinander synchronisiert werden. Dies geschieht, wenn eine Kante den Sender auf einem broadcast Channel besitzt und die anderen Kanten jeweils auf dem gleichen Channel einen Sender. Eine Kante mit solch einer Synchronisation auf einem broadcast Channel kann immer genommen werden (vorausgesetzt die Kante ist freigegeben), auch wenn nicht jede Kante freigegeben ist. Die Kanten, die freigegeben sind, werden jedoch genommen.
Die zweite Channel-Art wird *urgent Channel* genannt. Bei diesem darf keine Zeit vergehen, wenn die Kanten genommen werden können. Zusätzlich sind lediglich Datenbedingungen, jedoch keine Uhrenbedingungen an den Kanten erlaubt [26].

Des Weiteren besitzt eine Kante ein sogenanntes *Update-Label*. Dadurch ist es möglich, Variablen oder Uhren neue Werte zuzuweisen oder Funktionen auszuführen. In Abbildung 4 wird beispielsweise die Uhr `y` an der Kante `Aus` → `Dunkel` auf 0 gesetzt.

Auch in UPPAAL CORA können Kosten Zuständen und Kanten zugeordnet werden. Dies geschieht durch die integrierte Kostenvariable `cost`. In dem beschriebenen Beispiel kostet das Einschalten der Lampe beispielsweise 30 Energieeinheiten (`cost+=30`). Die Kosten in den Zuständen `Dunkel` bzw. `Hell` hingegen erhöhen sich stetig um eine Rate von 15 (`cost'==15`) bzw. 25 (`cost'==25`) Energieeinheiten.

3 Modellannahmen

Um unter anderem die Auswirkungen von smarten Geräten und einer Batterie auf die Maximierung des Eigenverbrauchs in verschiedenen Haushalten zu analysieren, werden zum einen vier verschiedene Modelle (konventionell/smart mit/ohne Batterie) und zum anderen drei unterschiedliche Haushalte in einer Nachbarschaft betrachtet. Diese besitzen jeweils eine PV-Anlage und sind an das öffentliche Stromnetz angeschlossen. Zudem ist jeder Haushalt mit einer Warmwasser-Wärmepumpe, einer Waschmaschine und einem Wäschetrockner ausgestattet, welche die betrachteten Geräte darstellen und in den smarten Modellen entsprechend smart sind. In den Modellen mit Batterie wird diese in den Haushalten als Energiespeicher genutzt.

Das Grundmodell eines Hauses ist ähnlich zu dem aus der Bachelorarbeit von Fabian Stein [11]. In Abbildung 5 wird ein Überblick über den Energiefluss in solch einem Haus mit Batterie gegeben.

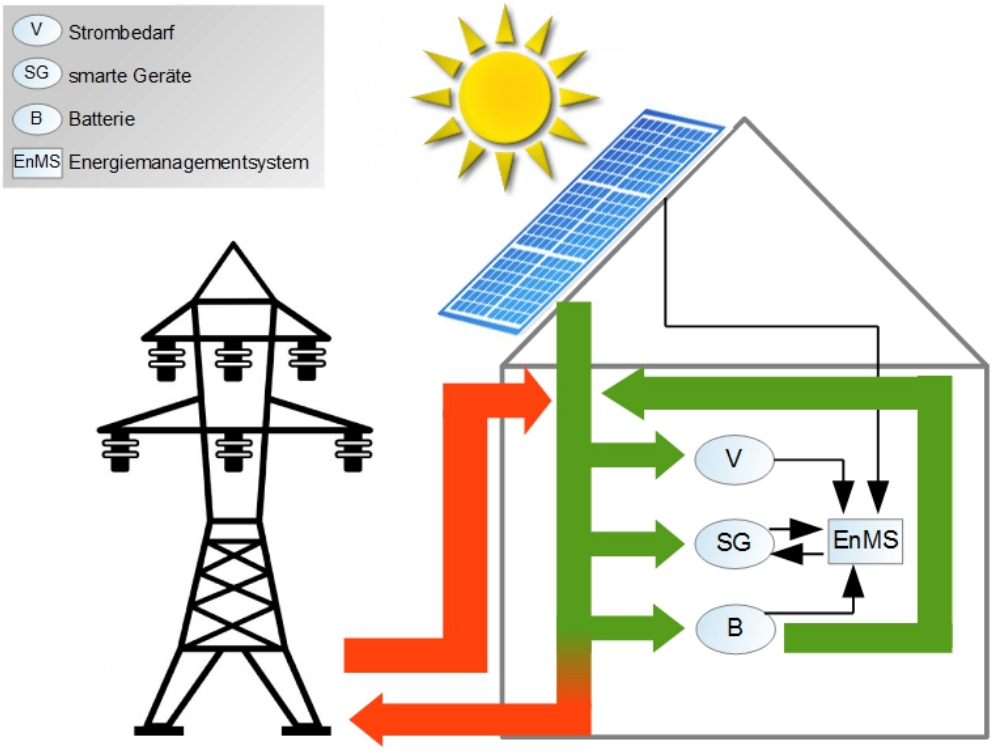


Abbildung 5: Energie- (rote und grüne Pfeile) und Informationsfluss (schwarze Pfeile) in einem Smart Home mit smarten Geräten und einer Batterie.

Der lokal produzierte Strom wird zuerst für den Stromverbrauch der konventionellen (V) und smarten Geräte (SG) genutzt. Besteht danach weiterhin ein Überschuss an lokal erzeugtem Strom, wird dieser in die Batterie (B) geladen, sofern diese genug freie Kapazität aufweist. Ist dies nicht der Fall, wird der restliche Strom in das öffentliche Stromnetz eingespeist. Produziert die PV-Anlage nicht genug Strom, um den Bedarf des Haushalts zu decken, wird, wenn möglich, Strom aus der Batterie genutzt. Andernfalls kann Strom aus dem öffentlichen Stromnetz bezogen werden. Es ist jedoch nicht möglich, dass die Batterie mit Strom aus dem öffentlichen Stromnetz geladen oder Strom aus der Batterie in dieses eingespeist wird.
Die grünen Pfeile stellen den Stromfluss dar, der die Maximierung des Eigenverbrauchs unterstützt, wohingegen die roten Pfeile den Austausch mit dem öffentlichen Stromnetz veranschaulichen, was sich nachteilig auf die Maximierung auswirkt.
Zudem kennzeichnen die schwarzen Pfeile den Informationsfluss zwischen den einzelnen Einheiten. Das Energiemanagementsystem (EnMS) erhält die Informationen über den Stromverbrauch, die Stromproduktion und den Ladezustand der Batterie. Außerdem kennt das System die Einschränkungen der smarten Geräte, sodass es den Schedule dieser und den Stromfluss im Haus dementsprechend steuern kann.

3.1 Haushaltsprofile

Um nicht zuletzt die Maximierung des Eigenverbrauchs in verschiedenen Haushalten beurteilen zu können, werden folgende Haushalte in einer Nachbarschaft betrachtet: eine Familie, ein Rentnerpaar und eine 26-40-jährige, alleinstehende, arbeitende Person. Die Familie besteht aus zwei Elternteilen, von denen ein Elternteil berufstätig ist, und zwei Kindern im Kindergarten- bzw. Grundschulalter.

Die Energieprofile der einzelnen Haushalte werden mit dem Tool ALPG berechnet, welches in Kapitel 2.1 vorgestellt wurde. Die Haushalte besitzen jeweils eine PV-Anlage und sind an das öffentliche Stromnetz angeschlossen, weisen jedoch kein Elektroauto oder Plug-in-Hybrid und nur in den vorgesehenen Modellen einen Batteriespeicher auf. Die Produktionsprofile der PV-Anlage basieren jeweils auf den Daten der Wetterstation in Twenthe für das Jahr 2014. Diese Daten werden vom Royal Nehterlands Meteorological Institute (KNMI) [27] zur Verfügung gestellt. Die PV-Anlagen sind jeweils mit einer Neigung von 35° nach Süden ausgerichtet und besitzen einen Wirkungsgrad von 17,5 %. Die Größe und somit auch die Stromproduktion der Anlage hängt vom jährlichen Strombedarf des betrachteten Haushalts ab und wird von ALPG berechnet. Ebenfalls lässt sich aus der Größe der PV-Anlage ihre Peakleistung in kWp berechnen. Diese gibt

die maximal mögliche Leistung einer PV-Anlage unter Standardbedingungen an. Für eine Leistung von 1 kWp sind etwa 8 m^2 nötig. [28], [29]
Jeder Haushalt ist zudem mit smarten und konventionellen Geräten ausgestattet, sodass sich der Strombedarf eines Haushalts aus beiden Typen zusammensetzt.

In Tabelle 4 ist für jeden Haushalt der ungefähre jährliche Gesamtstromverbrauch, sowie die jährliche annähernde Stromproduktion der PV-Anlage in kWh, ihre Größe in m^2 und ihre Peakleistung in kWp abgebildet.

	Gesamt-verbrauch (kWh)	**Produktion der PV-A.** (kWh)	**Größe der PV-A.** (m^2)	**Leistung der PV-A.** (kWp)
Familie	5100	4500	38	4,75
Rentner	3700	3200	27	3,4
Person	2600	1900	16	2

Tabelle 4: Ungefährer jährlicher Gesamtstromverbrauch des Haushalts, sowie die jährliche annähernde Stromproduktion der PV-Anlage, ihre Größe und ihre Peakleistung für jeden betrachteten Haushalt

3.2 Betrachtete Geräte

Als smarte Geräte werden solche bezeichnet, denen gewisse Daten übergeben werden, sodass diese Geräte aktiv sind, wenn der Überschuss an lokal erzeugtem Strom am höchsten ist, damit dieser bestmöglich genutzt werden kann.
Es werden eine Waschmaschine (WM), ein Wäschetrockner (TR) und eine Warmwasser-Wärmepumpe (WP) als smarte Geräte betrachtet.

Der Stromverbrauch eines typischen Waschgangs von 2 Stunden beträgt 0,98 kWh [30]. Für die betrachtete Familie bedeutet das einen jährlichen Stromverbrauch von zirka 327 kWh bei 334 Waschtagen im Jahr. Die Waschtage und der jährliche Stromverbrauch der Waschmaschine der anderen Haushalte sind Tabelle 5 zu entnehmen. In den smarten Modellen besitzt die Waschmaschine für jeden Waschgang einen Start- und Endzeitpunkt, sodass diese innerhalb dieses Intervalls wäscht, wenn der Überschuss an lokal erzeugtem Strom am höchsten ist.

Ein Trockenvorgang hat hingegen eine Stromverbrauch von 1,5 kWh [31] und eine Dauer von 1,5 Stunden. Die Anzahl der Trockentage in einem Jahr, sowie der jährliche Stromverbrauch des Trockners sind ebenfalls für jeden Haushalt in Tabelle 5 dargestellt. Bei

	Wasch-tage (Tage)	**Verbrauch der WM** (kWh)	**Trocken-tage** (Tage)	**Verbrauch des TR** (kWh)	**Warmwasser-verbrauch** (l)	**Verbrauch der WP** (kWh)
Familie	334	327	298	447	50344	679
Rentner	206	202	178	267	29316	468
Person	156	153	140	210	14997	325

Tabelle 5: Anzahl der Wasch- und Trockentage und der Warmwasserverbrauch in einem Jahr, sowie der jährlicher Stromverbrauch dieser Geräte für jeden betrachteten Haushalt

dem smarten Trockner ist es ähnlich wie bei der smarten Waschmaschine. Er besitzt ebenfalls einen Endzeitpunkt, zu dem der Trockenvorgang beendet sein muss. Da in den Modellen davon ausgegangen wird, dass die Wäsche direkt nach dem Waschvorgang in den Trockner gelegt wird, besitzt ein Trockenvorgang als Startzeitpunkt den Zeitpunkt, an dem der vorherige Waschgang der Waschmaschine beendet ist. Der Trockner kann somit nur nach einem Waschvorgang aktiv sein. Ob es nach einem Waschvorgang einen Trockenvorgang gibt, hängt zudem vom Wetter ab. Je sonniger das Wetter ist, desto unwahrscheinlicher ist es, dass der Trockner nach einem Waschvorgang die Wäsche trocknet.

Zusätzlich zu den untersuchten Haushalten, werden in den smarten Modellen vier verschiedene Intervalllängen der Waschmaschine bzw. des Trockners betrachtet, um deren Auswirkungen auf die Maximierung des Eigenverbrauchs zu analysieren. Das heißt, dass die Differenz zwischen dem Start- und Endzeitpunkt der Waschvorgänge, sowie die Differenz zwischen dem Endzeitpunkt der Waschvorgänge und dem Endzeitpunkt der Trockenvorgänge fest gesetzt werden. Die verschiedenen Intervalllängen werden in Intervallgruppen aufgeteilt. Diese sind in Tabelle 6 mit ihren Intervalllängen in Stunden dargestellt. So liegt beispielsweise in der zweiten Intervallgruppe die Intervalllänge der Waschmaschine bei 4 und die des Trockners bei 3 Stunden.

	Intervalllänge der Waschmaschine (h)	**Intervalllänge des Trockners** (h)
1. Intervallgruppe	3	2
2. Intervallgruppe	4	3
3. Intervallgruppe	6	4,5
4. Intervallgruppe	8	6

Tabelle 6: Betrachtete Intervallgruppen und ihre Intervalllängen der Wasch- bzw. Trockenvorgänge

Eine Warmwasser-Wärmepumpe erwärmt Wasser effizient und ist ausschließlich für die Erzeugung von Warmwasser zuständig. Sie ist unabhängig von der eigentlichen Heizung und nutzt die warme Raumluft und Strom, um das Wasser zu erhitzen [32]. Die Jahresarbeitszahl einer Warmwasser-Wärmepumpe gibt das Verhältnis von verwendeter Strommenge und ausgehender Wärmemenge über ein ganzes Jahr hinweg bei verschiedenen Betriebszuständen wieder. [33]
Die betrachtete Wärmepumpe hat ein Fassungsvermögen von 300 l und eine Jahresarbeitszahl von 3,4. Es werden dadurch 0,1 kWh Strom benötigt, um das Wasser um 1 °C zu erwärmen. [34]
Die Wassertemperatur sollte im Durchschnitt 55 °C betragen. In den konventionellen Modellen wird das Wasser schnellstmöglich auf diese Temperatur erhitzt, wohingegen in den smarten Modellen die Wassertemperatur zwischen 45 und 65 °C liegen kann. Das Wasser wird also möglichst dann erhitzt, wenn es einen Überschuss an lokal erzeugtem Strom gibt. Das Wasser verliert zum einen durch den Warmwasserverbrauch der Bewohner $(Warmwasserverbrauch\ in\ l) \cdot 0,1$ °C an Temperatur und zum anderen kontinuierlich 0,2 °C pro Stunde durch die Diskrepanz zur Außentemperatur.
In Tabelle 5 ist der jährliche Warmwasserverbrauch in Liter, sowie der jährliche Stromverbrauch der Warmwasser-Wärmepumpe in kWh für die betrachteten Haushalte dargestellt.

3.3 Batterie

Sonnige Tage haben oft zur Folge, dass die PV-Anlage mehr Strom produziert, als vom Haushalt benötigt wird. Um diesen überschüssigen Strom auch später für den Eigenverbrauch nutzen zu können, kann eine Batterie als Energiespeicher installiert werden. Der gespeicherte Strom kann genutzt werden, um den Strombedarf des Haushalts zu decken, wenn die PV-Anlage zu wenig Strom produziert. Somit ist es möglich, den Eigenverbrauch weiter zu steigern.
Es gibt zur Zeit zwei Batteriearten, um die PV-Anlage mit der Batterie zu kombinieren: Bleispeicher und Lithium-Ionen-Speicher. Die Hauptunterschiede liegen in der Lade- und Entladehäufigkeit, sowie in der Entladetiefe der Batterien. Der Bleispeicher kann weniger oft ent- und geladen und weniger tief entladen werden als der Lithium-Ionen-Speicher. Die Entladetiefe des Lithium-Ionen-Speicher liegt bei 70-100 %, wohingegen die des Bleispeichers nur bei 50-60 % liegt. Zudem ist der Energiewirkungsgrad und die Lebensdauer des Bleispeichers geringer [**?**]. Da die anfänglichen Investitionskosten für Lithium-Ionen-Speicher höher sind, folgt daraus, dass die durchschnittlichen Lebenszykluskosten beider Batteriearten in etwa übereinstimmen. [35]

In den betrachteten Modellen mit Batterie besitzen die Haushalte jeweils einen Lithium-Ionen-Speicher. Die nutzbare Kapazität der Batterie hängt von der Größe der PV-Anlage ab und wird anhand ihrer Peakleistung berechnet. Pro kWp PV-Leistung sollte 1 kWh nutzbare Kapazität vorhanden sein, um den Eigenverbrauch des lokal erzeugten Stroms optimal zu erhöhen. Somit ergibt sich für den Haushalt der Familie eine nutzbare Kapazität von 4,75 kWh, für den des Rentnerpaares 3,4 kWh und für den der alleinstehenden Person 2 kWh. Zudem liegt der Energiewirkungsgrad der betrachteten Batterien bei 95 % und somit der Speicherverlust beim Laden und Entladen der Batterie bei 5 %. [36]
Die Batterien werden in allen Modellen nur mit lokal produziertem Strom geladen und immer vollständig entladen.

4 Implementierung und Modellierung

Es wird das Tool ALPG genutzt, um die verschiedenen Energieprofile der betrachteten Haushalte zu erhalten (siehe Kapitel 3.1). Da dieses jedoch nicht alle benötigten Informationen liefert, wurde das Tool angepasst und erweitert. Dies wird in Kapitel 4.1 erläutert.
Da die von ALPG gelieferten Daten jedoch nicht kompatibel für UPPAAL CORA sind, müssen diese zuvor aufbereitet werden. Dird wird in Kapitel 4.2 beschrieben.
Für die smarten Modelle berechnet UPPAAL CORA anschließend den optimalen Schedule zur Maximierung des Eigenverbrauchs für die einzelnen Haushalte und betrachteten Intervallgruppen. Dies, sowie die Datenauslese aus UPPAAL CORA wird in Kapitel 4.3 geschildert.
In Kapitel 4.4 und Kapitel 4.5 wird erläutert, wie das konventionelle Modell und die Nutzung der Batterie außerhalb von UPPAAL CORA berechnet werden.
Abschließend werden alle Daten analysiert.

4.1 Anpassung des ALPG

ALPG liefert die verschiedenen Energieprofile der Haushalte. Es gibt den Stromverbrauch und die Stromproduktion eines Haushalts in Watt für jede Minute in einer CSV-Datei aus. Ebenso werden die Start- und Endzeitpunkte der Waschmaschine in Sekunden in einer Textdatei gespeichert.
Es fehlen jedoch die Endzeitpunkte des Wäschetrockners, sowie ein Wasserverbrauchsprofil, welches angibt, wie viel Wasser zu welcher Zeit verbraucht wird. Dies ist für die Simulation der Warmwasser-Wärmepumpe notwendig.
In der Bachelorarbeit von Fabian Stein [11] wird ein ähnliches Modell mit einer Waschmaschine, einem Wäschetrockner und einer Warmwasser-Wärmepumpe als smarte Geräte betrachtet. Da auch dort das Energieprofil des betrachteten Haushalts mit ALPG berechnet wurde, konnten die Anpassungen bezüglich des Wäschetrockners und der Wärmepumpe übernommen werden.
Die Endzeitpunkte des Trockners werden, wie die der Waschmaschine, in Sekunden in einer Textdatei gespeichert und liegen immer mindestens 2 Stunden hinter denen des Waschvorgangs. Somit gibt es genauso viele Trocken- wie Waschvorgänge.
Das Wasserverbrauchsprofil wird in einer CSV-Datei gespeichert. Es gibt pro Minute den Warmwasserverbrauch des Haushaltes in Liter an und hängt unter anderem von der Anwesenheit der Bewohner und ihren Aktivitäten ab (siehe [11, S.10]).

In Quellcode 1 ist der Programmausschnitt der Waschmaschine und des Trockners aus ALPG zu sehen. In der Funktion „endzeiten“(siehe Zeile 478-493) werden die Endzeitpunkte für die Waschmaschine und den Trockner gesetzt. Um die verschiedenen Intervalllängen der Waschmaschine bzw. des Trockners zu realisieren, werden die Endzeiten fest gesetzt. Anstatt einer zufällig berechneten Uhrzeit aus zwei gegebenen Uhrzeiten, werden die Intervalllängen und somit die Endzeitpunkte für die Waschmaschine und den Trockner fest definiert, beispielsweise in Quellcode 1 Zeile 482 und 486. Wie auch in [11] gibt es keine Endzeitpunkte zwischen 0 und 7 Uhr, da zu dieser Zeit die Bewohner schlafen. In Quellcode 1 Zeile 446-461 werden die Endzeiten für die verschiedenen Intervalle mithilfe der Funktion „endzeiten“berechnet. Die Ausgabe der Endzeitpunkte der Waschmaschine und des Trockners wurden so angepasst, dass diese für jede Intervallgruppe (siehe Tabelle 6) jeweils in einer Textdatei ausgegeben werden.
In den Zeilen 434-444 werden die Startzeiten der Waschmaschine gesetzt. Um die verschiedenen Intervallgruppen besser vergleichen zu können, haben alle Intervallgruppen die gleichen Startzeiten. Dabei wird darauf geachtet, dass der neue Waschvorgang frühestens nach dem letzten beginnt. Zudem ist aber auch gewährleistet, dass der nächste Endzeitpunkt der Waschmaschine größer als der letzte Endzeitpunkt des Trockners ist, damit der Trockner direkt nach dem Waschvorgang wieder genutzt werden kann. Damit dies für alle Endzeitpunkte gewährleistet ist, werden hierbei die Endzeitpunkte der Intervallgruppe mit den größten Intervalllängen betrachtet. Sind die Bedingungen für diese Endzeiten erfüllt, so auch für alle anderen.

```
420 class DeviceWashingMachine(TimeShiftableDevice):
421         def simulate(self, timeintervals, day, occupancy, washingMoment):
422                 washingtimeintervals = washingMoment - 30 + random.randint(0, 59)
433
434                 # Startzeiten: WaStart darf frühestens nach letzter WaEndzeit beginnen
                        und nächstes WaEnde muss größer gleich dem letzten TrEnde sein,
                    ansonsten wird es passend gesetzt. Es werden die Endzeiten des
                    Modells mit den größten Intervallen gewählt, da die Bedingungen
                    dann auch für die Modelle mit den kleineren Intervallen gewä
                    hrleistet sind
435
436                 if (len(self.dryerEndTimes6) > 0):
437                         if ((washingtimeintervals + 1440 * (day)) > self.
                                dryerEndTimes6[len(self.dryerEndTimes6) - 1] - min(
438                                         config.intervallTRmodell6, config.
                                             intervallWMmodell6) * 60):
439                                 self.StartTimes.append(washingtimeintervals + (1440 *
                                     (day)))
440                         else:
441                                 self.StartTimes.append(self.dryerEndTimes6[len(self.
                                     dryerEndTimes6) - 1] - min(config.
                                     intervallTRmodell6, config.intervallWMmodell6) *
```

```
                                    60 + 60)
442                                 washingtimeintervals = ((self.dryerEndTimes6[len(self.
                                        dryerEndTimes6) - 1] - min(config.
                                        intervallTRmodell6, config.intervallWMmodell6) *
                                        60) % 1440) + 60
443                 else:
444                         self.StartTimes.append(washingtimeintervals + (1440 * (day)))
445
446                 # Endzeiten für verschiedene Intervalle
447                 endzeiten, trocknerendzeiten = self.endzeiten(washingtimeintervals,
                        day, config.intervallWMmodell3, config.intervallTRmodell3)
448                 self.EndTimes3.append(endzeiten)
449                 self.dryerEndTimes3.append(trocknerendzeiten)
450
451                 endzeiten, trocknerendzeiten = self.endzeiten(washingtimeintervals,
                        day, config.intervallWMmodell4, config.intervallTRmodell4)
452                 self.EndTimes4.append(endzeiten)
453                 self.dryerEndTimes4.append(trocknerendzeiten)
454
455                 endzeiten, trocknerendzeiten = self.endzeiten(washingtimeintervals,
                        day, config.intervallWMmodell5, config.intervallTRmodell5)
456                 self.EndTimes5.append(endzeiten)
457                 self.dryerEndTimes5.append(trocknerendzeiten)
458
459                 endzeiten, trocknerendzeiten = self.endzeiten(washingtimeintervals,
                        day, config.intervallWMmodell6, config.intervallTRmodell6)
460                 self.EndTimes6.append(endzeiten)
461                 self.dryerEndTimes6.append(trocknerendzeiten)
477
478         #berechnet die Endzeiten der Waschmaschine und des Trockners am Tag 'day'. '
                washingtimeintervals' ist die Startzeit der Waschmaschine an dem Tag. '
                intervallWM' und 'intervallTR' sind die Intervalllängen der Waschmaschine
                und des Trockners
479         def endzeiten(self, washingtimeintervals, day, intervallWM, intervallTR):
480                 # Endtime is after intervalllaengeWM
481                 if washingtimeintervals < (24 - intervallWM - intervallTR) * 60:
482                         help = washingtimeintervals + intervallWM * 60
483                         endTimes = 1440 * (day) + help
484                         dryerEndTimes = 1440 * day + help + intervallTR * 60
485                 elif (washingtimeintervals < (24 - intervallWM) * 60):
486                         help = washingtimeintervals + intervallWM * 60
487                         endTimes = 1440 * day + help
488                         dryerEndTimes = 1440 * (day + 1) + 8*60
489                 else:
490                         help = 8*60
491                         endTimes = 1440 * (day + 1) + help
492                         dryerEndTimes = 1440 * (day + 1) + help + intervallTR * 60
493                 return endTimes, dryerEndTimes
```

Quellcode 1: Modifikationen der Waschmaschine in ALPG (devices.py).

Da ALPG für viele Werte Zufallszahlen verwendet, wurden einige Parameter fest gesetzt, um genaue Aussagen über die Haushalte treffen und die Ergebnisse beurteilen

zu können. Für jeden betrachteten Haushalt wurde die Personenanzahl und der jährliche Stromverbrauch festgelegt. Dies ist zum Beispiel für die Familie in Quellcode 2 Zeile 504 und 505 zu sehen, wobei in Quellcode 2 der Programmcode des betrachteten Haushalts der Familie dargestellt ist.
Bei dem Haushalt der Familie wurde zudem das Alter der Kinder so abgestimmt, dass es ein schulpflichtiges Kind und ein Kindergartenkind gibt. Dies wurde in Quellcode 2 in den Zeilen 527-531 realisiert.

```
501 class HouseholdFamilySingleWorker(Household):
502         def __init__(self, parttime):
503                 self.generate()
504                 numKids = 2
505                 self.ConsumptionYearly=3360+(700*numKids)+450
513
514                 #geaendert, da 2. Person nicht arbeiten soll
515                 self.Persons.append(persons.PersonWorker(ageParents))
516                 self.Persons[1].generateWorkdays(1)
526
527                 #now add the kids
528                 self.Persons.append(persons.PersonStudent(random.randint(3,5)))
                                #Kita-Kind
529                 self.Persons.append(persons.PersonStudent(random.randint(7,10)))#
                        schulpflichtiges Kind
530                 #Zeitpunkt, wann Kita-Kind wiederkommt
531                 self.Persons[2].WorkdayArrival_Avg=self.Persons[2].WorkdayLeave_Avg+
                        profilegentools.gaussMinMax(4.5*60,30)
```

Quellcode 2: Modifikationen an der Klasse der Familie in ALPG (households.py).

4.2 Datenextrahierung

Da die von ALPG gelieferten Daten nicht das passende Format für das Modell in UPPAAL CORA haben, müssen diese noch bearbeitet werden.
Auch in der Bachelorarbeit von Fabian Stein [11] mussten die Daten aufbereitet werden. Dies wurde mit Powershell, einem Kommandozeileninterpreter von Microsoft, realisiert. In dieser Arbeit hingegen wurde die objektorientierte Skriptsprache Python gewählt, da diese für eine Vielzahl von Plattformen (Unix/Linux, Windows, MacOS, etc.) kostenlos verfügbar ist [37] und zudem das Tool selber ebenfalls in Python implementiert ist. Dennoch wurden einige Daten ähnlich bearbeitet.

Zuerst müssen die Stromverbrauchs- und -produktionsdaten angepasst werden. Dies geschieht ähnlich wie in [11]. Da diese Daten in Watt für jede Minute von ALPG ausgegeben werden, werden diese so aufbereitet, dass sie in Wh für jeden Zeitschritt

bzw. jede Zeiteinheit bereitgestellt werden. Die Zeitschritte beziehen sich auf jene aus dem UPPAAL-CORA-Modell aus Kapitel 4.3.1, welche in diesem Fall halbstündliche Zeiteinheiten sind. Es wird für jeden Zeitschritt der Durchschnitt der Daten berechnet und in Wh umgerechnet, wie beispielsweise in Quellcode 3 für die Produktionsdaten zu sehen.

```
45      #liest aus ALPG die PV-Produktionsdaten aus und verarbeitet diese, sodass pro
            Zeiteinheit die PV-Produtkion in Wh gespeichert wird
46      def pvProduktion(self):
58
59          #da die PV-Daten minutenweise in W in ALPG ausgegeben werden, wird der
                Durschnitt für jede Zeiteinheit berechnet
60          for zeitschritt in range(tagesAnzahl*zeitschritteProTag):
61              zeitschrittDaten = pvDaten[int(zeitschritt*(24/zeitschritteProTag)*60):int
                    ((zeitschritt+1)*(24/zeitschritteProTag)*60)]
62              durchschnitt = numpy.mean(zeitschrittDaten)
63
64              #in Wh umrechnen
65              zeitschrittErzeugnis = int(round((-1)*durchschnitt*(24/zeitschritteProTag)
                    ,0))
66              pvProduktion.append(zeitschrittErzeugnis)
67              pvProduktioncsv.append([zeitschrittErzeugnis])
```

Quellcode 3: Aufbereitung der Produktionsdaten aus ALPG (Datenextrahierung.py)

Ähnlich der Implementierung in Powershell von Fabian Stein [11] ist die Bearbeitung der Warmwasserdaten in Quellcode 4. Da von ALPG der minütliche Warmwasserverbrauch in l ausgegeben wird, wird in Zeile 185 und 186 für jede Zeiteinheit die Summe der minütlichen Daten berechnet, sodass der Warmwasserverbrauch in l pro Zeitschritt bereitgestellt wird.

```
169     # liest aus ALPG die Warmwasserdaten aus und verarbeitet diese, sodass pro
            Zeiteinheit der Wasserbrauch in Liter gespeichert wird
170     def boVerbrauch(self):
182
183         #da der Wasserverbrauch minutenweise in ALPG ausgegeben wird, wird für jede
                Zeiteinheit die Summe gebildet
184         for zeitschritt in range(tagesAnzahl*zeitschritteProTag):
185             zeitschrittDaten = wasserDaten[int(zeitschritt*(24/zeitschritteProTag)*60)
                    :int((zeitschritt+1)*(24/zeitschritteProTag)*60)]
186             summe = int(round(numpy.sum(zeitschrittDaten)))
187             wasserVerbrauch.append(summe)
188             wasserVerbrauchcsv.append([summe])
```

Quellcode 4: Aufbereitung der Warmwasserverbrauchsdaten aus ALPG (Datenextrahierung.py)

Um die Auswirkungen der Intervalllängen der Wasch- bzw. Trockenvorgänge im Vergleich zum konventionellen Modell besser beurteilen zu können, wird zu den anderen Modellen noch ein weiteres untersucht. In diesem Modell sind die Waschmaschine und der Trockner smart, die Wärmepumpe jedoch nicht. Das bedeutet, dass die Wassertemperatur möglichst auf 55 °C gehalten wird. Dazu wird in Quellcode 5 erst der Stromverbrauch der konventionellen Wärmepumpe berechnet (siehe Zeile 146-147) und anschließend der Stromverbrauch der konventionellen Geräte inklusive dem der Wärmepumpe in einer Textdatei gespeichert (siehe Zeile 148).

```
134     #berechnet den Stromverbrauch der nicht-smarten Geräte und der Wärmepumpe unter
            der Annahme, dass diese nicht-smart ist
135     def stromVerbrauchBoNs(self, stromverbrauch, boVerbrauch):
143
144         for i in range(0, tagesAnzahl * zeitschritteProTag):
145             # Temperatur wird neu berechnet. Temperatur wird immer versucht auf 55
                    Grad (temperaturNS) zu halten. Es kann aber nicht mehr als 2 Grad (mit
                    200W) erhöht werden
146             temperatur = temperatur + temperaturdelta / 10 - boVerbrauch[i] / 10
147             delta = min(200, temperaturNS * 100 - temperatur * 100)
148             stVerbrauchBoNS.append(round(delta+stromverbrauch[i]))
149             temperatur = temperatur + delta / 100
```

Quellcode 5: Aufbereitung der Verbrauchsdaten der konventionellen Geräte inklusive der der Wärmepumpe (Datenextrahierung.py)

Für die Start- und Endzeitpunkte der Waschmaschine und des Trockners liefert ALPG sekündliche Daten. Diese werden, wie auch in der Bachelorarbeit von Fabian Stein [11], in die Zeiteinheit umgerechnet. Für die Waschmaschine ist dies in Quellcode 6 dargestellt.

```
214     #liest aus ALPG die Waschmaschinenstart- und -enddaten aus und verarbeitet diese,
            sodass die Start- bzw Enddaten in Zeiteinheiten angegeben werden
215     def waVerbrauch(self, pfad_out2, dateiname_Start, dateiname_Ende):
242
243         #da ALPG die Daten in Sekunden angibt, werden die Daten in die Zeiteinheit
                umgerechnet und in einer Liste
244         #gespeichert
245         for i in waschmaschineStartlang:
246             #t=i.split()
247             waschmaschineStart.append(int(round(float(i)/(60*60*(24/zeitschritteProTag
                    )),0)))
248             waschmaschineStartcsv.append([int(round(float(i)/(60*60*(24/
                    zeitschritteProTag)),0))])
249         for i in waschmaschineEndlang:
```

```
250             #t=i.split()
251             waschmaschineEnd.append(int(round(float(i)/(60*60*(24/zeitschritteProTag))
                    ,0)))
252             waschmaschineEndcsv.append([int(round(float(i)/(60*60*(24/
                    zeitschritteProTag)),0))])
```

Quellcode 6: Aufbereitung der Waschmaschinendaten aus ALPG (Datenextrahierung.py)

In Quellcode 7 ist die Implementierung des Trockners zu sehen.
Anders als in der Bachelorarbeit von Fabian Stein [11], in der einige Endzeitpunkte des Trockners in Abhängigkeit der PV-Produktion nach der Berechnung des optimalen Schedules mit UPPAAL CORA entfernt werden, werden in dieser Arbeit diese Endzeitpunkte bereits vor der Berechnung in Abhängigkeit der Solareinstrahlung entfernt. Dies hat unter anderem den Vorteil, dass die entfernten Endzeitpunkte keinen Einfluss auf den optimalen Schedule haben. Zudem ist durch die Nutzung der Solareinstrahlung das Entfernen unabhängig von der Ausrichtung und der Größe der PV-Anlage.
Zur Berechnung dieser Trocknerdaten wird die ALPG-Datei solarirradiation.csv, die die Solareinstrahlung der betrachteten Nachbarschaft pro Stunde für ein ganzes Jahr beinhaltet, eingelesen. Diese Daten werden zur Verwendung in Quellcode 7 Zeile 307-309 auf die Zeitschritte umgerechnet. Die Trockneraktivität wird anschließend in den Zeilen 323-345 in Abhängigkeit der Solareinstrahlung in den nächsten 24 Stunden nach Startzeitpunkt der Waschmaschine (Zeile 311-313) berechnet. Somit ist der Trockner in den verschiedenen Intervallgruppen an den gleichen Tagen aktiv. Je sonniger es ist, desto wahrscheinlicher ist es, dass die Wäsche draußen getrocknet und der Trockner nicht benutzt wird. So wird dieser beispielsweise an sehr sonnigen Tagen nur zu 20 % (Zeile 343) und an sehr bedeckten Tagen zu 90 % (Zeile 325) genutzt. Die Aktivität wird als binäre Liste gespeichert. Dabei stellt eine 1 die Aktivität des Trockners nach dem Waschvorgang dar (siehe z. B. Quellcode 7 Zeile 326 und 329) und eine 0 die Inaktivität (siehe Quellcode 7 Zeile 319-321).
In Quellcode 7 Zeile 357-401 werden die Endzeitpunkte des Trockners aus ALPG in die Zeiteinheit umgerechnet und mit der Trockneraktivität abgeglichen. Es werden die Zeitpunkte entfernt, an denen der Trockner nicht aktiv ist.

```
294     # berechnet in Abhängigkeit der Solareinstrahlung die Aktivität des Trockners
295     def trockneraktiv(self, waschmaschineStart):
306
307         # da ALPG die Daten stündlicht angibt, werden diese auf die Zeiteinheit
                umgerechnet
308         for i in range(0, math.floor(len(solareinstrahlung) * zeitschritteProTag/24)):
309             einstrahlung.append(int(round(solareinstrahlung[math.floor(i / (
                    zeitschritteProTag/24))] / (zeitschritteProTag/24))))
310
311         # berechnet die Sonneneinstrahlung einen Tag nach Waschmaschinenstart
```

```
312         for i in range(0, len(waschmaschineStart)-1):
313             sonne.append(numpy.sum(einstrahlung[waschmaschineStart[i]:
                    waschmaschineStart[i]+zeitschritteProTag-1]))
318
319         for i in range(0, len(waschmaschineStart)):
320             trockneraktiv.append(0)
321             trockneraktivcsv.append([0])
322
323         # es wird berechnet, ob der Trockner nach dem Waschvorgang trocknet oder
                aufgrund der hohen Sonneneinstrahlung nicht. In trockneraktiv wird
                gespeichert, ob der Trockner nach dem Waschvorgang aktiv (1) ist oder
                nicht (0)
324         for t in range(0, len(waschmaschineStart)):
325             if (sonne[t] <= 500 and random.random() <= 0.9):
326                 trockneraktiv[t] = 1
327                 trockneraktivcsv[t] = [1]
328             elif (sonne[t] <= 1000 and random.random() <= 0.7):
329                 trockneraktiv[t] = 1
330                 trockneraktivcsv[t] = [1]
331             elif (sonne[t] <= 1500 and random.random() <= 0.6):
332                 trockneraktiv[t] = 1
333                 trockneraktivcsv[t] = [1]
334             elif (sonne[t] <= 2000 and random.random() <= 0.5):
335                 trockneraktiv[t] = 1
336                 trockneraktivcsv[t] = [1]
337             elif (sonne[t] <= 2500 and random.random() <= 0.4):
338                 trockneraktiv[t] = 1
339                 trockneraktivcsv[t] = [1]
340             elif (sonne[t] <= 3000 and random.random() <= 0.3):
341                 trockneraktiv[t] = 1
342                 trockneraktivcsv[t] = [1]
343             elif (sonne[t] > 3000 and random.random() <= 0.2):
344                 trockneraktiv[t] = 1
345                 trockneraktivcsv[t] = [1]
356
357     # liest aus ALPG die Trocknerenddaten aus und verarbeitet diese, sodass die
            Enddaten in Zeiteinheiten angegeben werden
358     def trVerbrauch(self, pfad_out2, dateiname_Ende, trockneraktiv):
377
378         for i in trocknerlang:
379             trockner.append(int(round(float(i)/(60*60*(24/zeitschritteProTag)),0)))
380             trocknercsv.append([int(round(float(i)/(60*60*(24/zeitschritteProTag)),0))
                    ])
396
397         #es wird gespeichert, ob der Trockner nach dem Waschvorgang trocknet in Abhä
                ngigkeit der hohen Sonneneinstrahlung (trockneraktiv)
398         for t in range(0,len(trockner)):
399             if trockneraktiv[t] == 1:
400                 neuerTrockner.append(trockner[t])
401                 neuerTrocknercsv.append([trockner[t]])
```

Quellcode 7: Aufbereitung der Trocknerdaten aus ALPG (Datenextrahierung.py)

4.3 Smartes Modell

In den smarten Modellen wird das Ziel, den Eigenverbrauch zu maximieren, unter anderem durch die modifizierten Haushaltsgeräte (Waschmaschine, Wäschetrockner, Warmwasser-Wärmepumpe) realisiert. Die Variabilität und damit die Effektivität eines Gerätes hängt dabei von verschiedenen Faktoren ab: von den natürlichen Grenzen (z.B. die Einschränkung der Wassertemperatur einer Warmwasser-Wärmepumpe), von den Nutzern (z.B. der Endzeitpunkt einer Waschmaschine) und schließlich von anderen Geräten (z.B. der Beendigung des Waschvorgangs vor dem Trockenvorgang) [24, S.5]. Der optimale Schedule dieser Geräte wird mithilfe von UPPAAL CORA berechnet. Dazu werden die smarten Geräte in Kapitel 4.3.1 in diesem Tool modelliert. Anschließend wird mit dem Kommandozeilen-Verifizierer *verifyta* der optimale Schedule berechnet und in einer Datei gespeichert (siehe Kapitel 4.3.2). Diese Daten werden daraufhin in Python ausgelesen und bearbeitet, wie in Kapitel 4.3.3 beschrieben.

4.3.1 Modellierung in UPPAAL CORA

Um den Eigenverbrauch in den smarten Modellen zu maximieren, wird der optimale Schedule mit UPPAAL CORA berechnet. Dazu werden die smarten Geräte mithilfe von LPTAs in UPPAAL CORA modelliert und zu einem System kombiniert, sodass eine möglichst realitätsnahe Abbildung eines Haushalts mit smarten Geräten gegeben ist.
Dies wurde bereits in dem Paper [24] und der Bachelorarbeit von Fabian Stein [11] realisiert. Allerdings war es aufgrund der Komplexität in diesen Arbeiten nicht möglich, mehr als zwei Tage von UPPAAL CORA berechnen zu lassen, trotz des Festlegens der Zeiteinheit auf eine halbe Stunde. In [11] konnte zwar ein ganzes Jahr analysiert werden, jedoch waren dafür mehrere Schritte nötig. Zuerst wurden Daten für zwei Tage eingelesen, sodass der optimale Schedule für diese Tage berechnet werden konnte. Anschließend wurden die erzeugten Daten wieder ausgelesen und neue Daten eingelesen. Dieser Vorgang wurde 188 mal durchgeführt, um den optimalen Schedule für ein Jahr zu erhalten.
Diese Vorgehensweise hat jedoch den Nachteil, dass bei der Berechnung des optimalen Schedules nicht alle Tage gleichzeitig mit einberechnet werden können. So ist es bei der Berechnung des zweiten Tages beispielsweise nicht möglich, den dritten Tag mit einzubeziehen. Dies kann möglicherweise zu einem verfälschten Schedule führen.
Um dies zu verhindern, wurde in Zusammenarbeit mit Rom Langerak und Stefano Schivo von der Universität in Twente das Modell so optimiert, dass es möglich ist, ein ganzes Jahr zu betrachten, wenn die Zeiteinheit auf eine halbe Stunde festgelegt ist.

In Abbildung 6 ist der Automat der Waschmaschine (WM) dargestellt. Diese ist mindestens so lange nicht in Betrieb (`kein_Betrieb`) bis sie beladen wird (`x>=wmstart[wmi]`). Zudem muss die Waschmaschine spätestens zwei Stunden (Waschdauer `wmperiode`) vor dem Endzeitpunkt (`wmende[wmi]`) in Betrieb gehen (Wechsel in `Betrieb`). Dies ist durch die Invariante `x<=wmende[wmi]-wmperiode` gewährleistet. Die Waschmaschine läuft zwei Stunden (`wmuhr<=wmperiode`) und informiert dann den Wäschetrockner über den Channel `update`, dass der Waschvorgang beendet ist. Anschließend ist sie wieder bereit für den nächsten Auftrag (Wechsel in `kein_ Betrieb`).
Bei einer Umsetzung in die Realität informiert die Waschmaschine nicht den Trockner, sondern den Hausbewohner, etwa über eine Smartphone-Applikation. Dieser kann daraufhin, wenn gewünscht, den Trockner beladen und anstellen kann.
Der Automat der Waschmaschine wurde im Vergleich zu dem Automaten in dem Paper [24] und der Bachelorarbeit [11], der in Abbildung 7 abgebildet ist, derart optimiert, dass der Zustand `Auftrag` entfernt und die Betriebszeit (Integer `zeit`) durch eine Uhr (`wmuhr`) ersetzt werden konnte. Dies verringert die Komplexität des Automaten deutlich, da es weniger Zustandswechsel gibt und die Uhr nach jedem Waschvorgang zurückgesetzt wird.

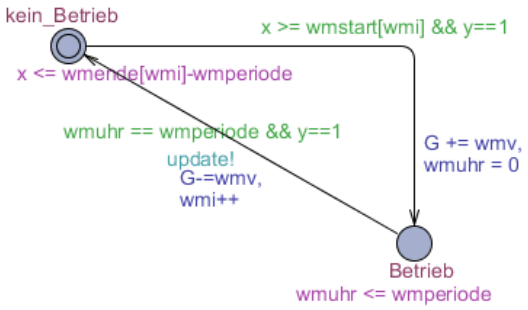


Abbildung 6: Neuer Automat der Waschmaschine.

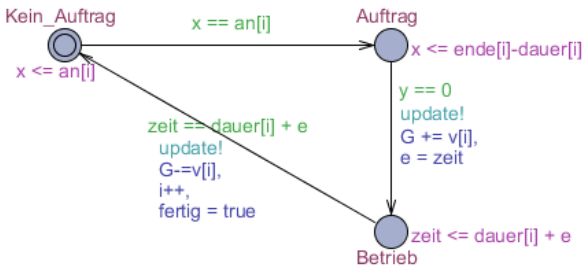


Abbildung 7: Alter Automat der Waschmaschine aus [24] bzw. [11].

Ebenso konnte der Automat des Trockners verbessert werden, indem die Betriebszeit durch eine Uhr ersetzt wurde. Da der Trockner, wie in Kapitel 4.2 beschrieben, nicht immer nach einem Waschvorgang trocknet, was in dem Paper [24] und der Bachelorarbeit [11] der Fall ist, musste der Automat dementsprechend angepasst werden.
Der angepasste Automat des Trockners ist in Abbildung 8 dargestellt. Nachdem die Waschmaschine ihren Waschvorgang beendet und den Wäschetrockner informiert hat, bekommt dieser entweder einen Auftrag (Wechsel in `Auftrag`) oder muss auf den nächsten Waschgang warten (Verbleib in `kein_ Auftrag`). Letzteres ist der Fall, wenn die

Möglichkeit besteht, dass der neue Waschvorgang vor dem Trockenvorgang beendet ist. Dies ist durch den Guard `ende[i]>=wmstart[wmi]+intTR` gegeben, wobei `intTR` die Intervalllänge eines Trockenvorgangs angibt. Der restliche Automat ist ähnlich zu dem der Waschmaschine aufgebaut.

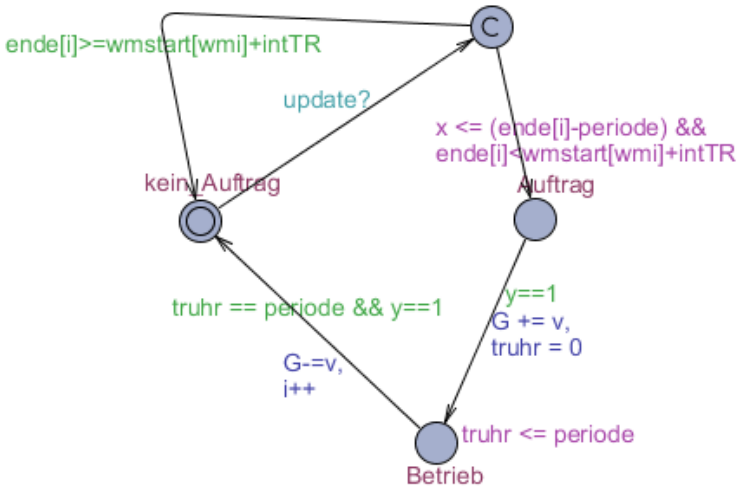


Abbildung 8: Automat des Trockners.

Abbildung 9 zeigt den Automaten der Warmwasser-Wärmepumpe (WP). Dieser ist ähnlich zu dem Automaten aus [24] bzw. [11]. Die Wärmepumpe erhält ihre Variabilität dadurch, dass die Warmwassertemperatur nicht gleichbleibend sein muss, sondern zwischen 45 °C und 65 °C schwanken darf. Sie kann entweder das Wasser erhitzen (`An`) oder ausgeschaltet sein (`Aus`), solange die Temperaturgrenzen nicht verletzt werden. Dies ist durch die Invarianten der Zustände gegeben.

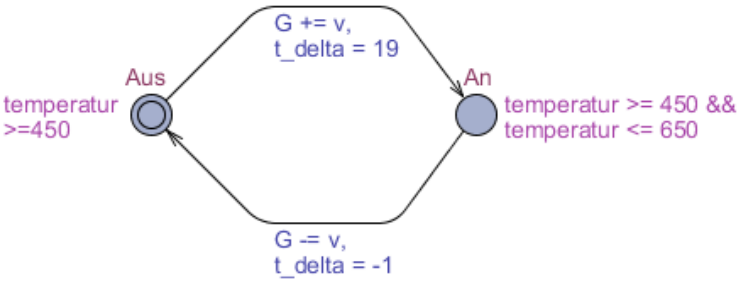


Abbildung 9: Automat der Warmwasser-Wärmepumpe.

In Abbildung 10 ist das Modell des Controllers dargestellt. Er dient als Schnittstelle zum Datenmodell und aktualisiert die Stromproduktion der PV-Anlage und den

Stromverbrauch des Haushalts über die Zeit. Dies geschieht jeweils durch eine Abfrage (`R()`) zu jeder Zeiteinheit (`y==1`). Diese Abfrage berechnet den Reststrom, d. h. die Differenz des lokal produzierten Stroms und des Gesamtstromverbrauchs inklusive des Verbrauchs der smarten Geräte. In dieser Abfrage sind somit ebenso die Verbrauchsdaten der smarten Geräte, die in den Automaten der smarten Geräte gesetzt werden, enthalten. Dadurch ist der Controller gleichzeitig mit den smarten Geräten verbunden. Auch der Controller wurde in Zusammenarbeit mit der Universität Twente verbessert. Der sogenannte `Ticker` aus dem Controller aus Abbildung 11 von [24] bzw. [11] konnte entfernt werden. Es reicht aus, dass der Reststrom einmal pro Zeiteinheit aktualisiert wird und nicht jedes mal, wenn ein smartes Gerät an- oder ausgeschaltet wird. Dies genügt, da die Kosten auch nur einmal pro Zeiteinheit erhöht werden. Diese Änderungen führen dazu, dass der Controller nur noch aus einem Zustand besteht.

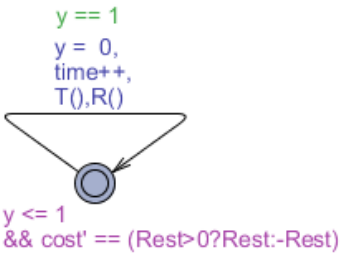


Abbildung 10: Neuer Automat des Controllers.

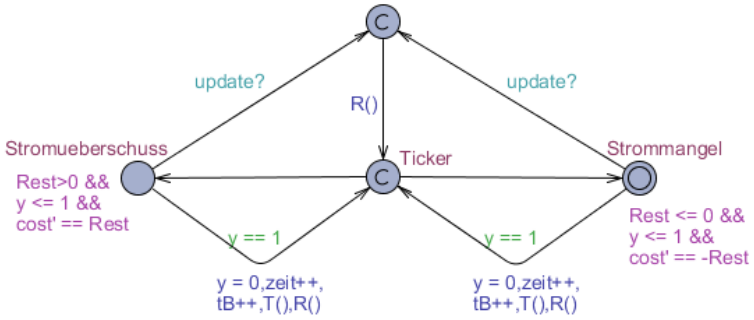


Abbildung 11: Alter Automat des Controllers aus [24] bzw. [11].

Das Ziel ist es den Eigenverbrauch des lokal erzeugten Stroms zu maximieren. Dabei lässt sich der Eigenverbrauch wie folgt berechnen:

$$Eigenverbrauch = \frac{Eigennutzung}{Stromproduktion} = \frac{Stromproduktion - Stromverkauf}{Stromproduktion} = 1 - \frac{Stromverkauf}{Stromproduktion}.$$

Die gestellte Minimierungsaufgabe an das smarte Modell besteht somit aus zwei Komponenten: zum einen möglichst wenig Strom aus dem öffentlichen Stromnetz zu beziehen, also einzukaufen, zum anderen möglichst wenig Strom in dieses einzuspeisen und somit zu verkaufen. Der lokal erzeugte Strom sollte also möglichst für den Eigenverbrauch genutzt werden. Die Kosten werden folglich durch die Summe des Stromein- und Stromverkaufs dargestellt, sodass sich folgende Minimierungsfunktion ergibt:

$$mincost(l) = inf\{cost\} = inf\{\sum(Stormeinkauf + Stromverkauf)\}.$$

l beschreibt hier die Anzahl der betrachteten Zeitschritte, also den Wert der Uhr `x`.
Im Modell werden der Stromein- und Stromverkauf durch die Funktion „$R()$“dargestellt,

die den Reststrom (`Rest`) berechnet. Ein negativer Wert steht für den Stromeinkauf und ein positiver für den Stromverkauf. Die Minimierungsaufgabe wird im Controller umgesetzt. Der eingekaufte bzw. verkaufte Strom wird in jeder Zeiteinheit in dem Zustand des Controllers zu den Kosten addiert (`cost'==(Rest>0?Rest:-Rest)`).

Durch die Verbesserung der Automaten in Zusammenarbeit mit Rom Langerak und Stefano Schivo von der Universität Twente konnte die Komplexität so verbessert werden, dass UPPAAL CORA den optimalen Schedule in einem Durchlauf für ein ganzes Jahr berechnen kann, wenn von einer Zeiteinheit von einer halben Stunde ausgegangen wird. Daher müssen die Verbrauchs- und Produktionsdaten nicht jedes mal aus- und wieder eingelesen, sondern nur einmal für jeden Haushalt als Parameter in das Modell kopiert werden.

4.3.2 Berechnung des optimalen Schedules

Um den optimalen Schedule mithilfe von UPPAAL CORA zu berechnen, wird der Kommandozeilen-Verifizierer *verifyta* genutzt (siehe Kapitel 2.2.1). Durch den Konsolenbefehl aus Quellcode 8 wird der beste Pfad mit der Query „$E <> x == 17522$“, die in „Query17522.q“gespeichert ist, für 17522 Zeitschritte für das angegebene Modell berechnet. Dies entspricht der Berechnung für ein ganzes Jahr. Der berechnete beste Pfad wird in einer Textdatei gespeichert.

```
verifyta -t3 FamilieModell4.xml ..\Query17522.q 2> Modell4.txt
```

Quellcode 8: Konsolenbefehl zur Berechnung des optimalen Schedule

4.3.3 Datenauslese

Um den in der entsprechenden Datei gespeicherten optimalen Schedule analysieren zu können, muss dieser zunächst ausgelesen werden. Dies wird, wie auch die Datenextrahierung in Kapitel 4.2, in Python realisiert und ist in Quellcode 9 dargestellt. Dazu wird die entsprechende Datei in Quellcode 9 Zeile 79-83 in Python eingelesen und für jeden Zeitschritt ($Delay : 1$) ermittelt, welches smarte Gerät zu diesem Zeitpunkt aktiv ist. Ist ein Gerät aktiv, so wird der Stromverbrauch dieses Geräts in die entsprechende Liste gespeichert (siehe Quellcode 9 Zeile 98, 103, 108). Ist es inaktiv, so wird eine 0 der Liste hinzugefügt (siehe Quellcode 9 Zeile 100, 105, 110).
Somit wird für jedes smarte Gerät eine Liste mit dessen Stromverbrauch pro Zeitschritt erzeugt. Diese stellen den optimalen Schedule dar.

```
8  #Stromverbrauch der "smarten" Geräte in Wh pro Zeiteinheit
9  wmV = 245
10 trV = 500
11 boV = 200
74
75 #Liest die UPPAAL-Daten aus der entsprechenden txt-Datei aus und verarbeitet diese.
76 class Auslesen:
77     #Gibt den Verbrauch der smarten Geräte in Listen zurück
78     def auslesen(batchpfad):
79         auslese = []
80         f = open(batchpfad, 'r', encoding='utf-8')
81         for line in f:
82             auslese.append(line)
83         f.close()
84         auslese_sortiert = []
85
86         #Es werden die dritten Zeilen über "Delay: 1" betrachtet und geschaut, ob dort
               die jeweiligen Geräte aktiv sind.
87         for idx, val in enumerate(auslese):
88             if 'Delay: 1' in val:
89                 auslese_sortiert.append(auslese[idx - 3])
90         auslese_sortiert.pop(0)
91         auslese_sortiert.pop(len(auslese_sortiert)-1)
92
93         wmVerbrauch = []
94         trVerbrauch = []
95         boVerbrauch = []
96         for idx, val in enumerate(auslese_sortiert):
97             if 'WM. Betrieb' in val:
98                 wmVerbrauch.append(wmV)
99             else:
100                wmVerbrauch.append(0)
101
102            if 'TR. Betrieb' in val:
103                trVerbrauch.append(trV)
104            else:
105                trVerbrauch.append(0)
106
107            if 'WP. An' in val:
108                boVerbrauch.append(boV)
109            else:
110                boVerbrauch.append(0)
111
112        return wmVerbrauch, trVerbrauch, boVerbrauch
```

Quellcode 9: Auslese und Verarbeitung des von UPPAAL-CORA berechneten optimalen Schedule (Auslesen.py)

4.4 Konventionelles Modell

In den konventionellen Modellen gibt es nur einen vorgegebenen Schedule, da die betrachteten Geräte nicht smart sind. Die Waschmaschine wäscht somit direkt nach der Auftragserteilung, der Wäschetrockner trocknet direkt nach Beendigung des Waschvor-

gangs, vorausgesetzt es gibt einen Trockenvorgang, und die Warmwasser-Wärmepumpe hält die Temperatur möglichst auf 55 °C.
Der Schedule bzw. der Stromverbrauch der betrachteten Geräte wurde in Python implementiert und ist in Quellcode 10 zu sehen. Dazu werden die Startzeiten der Waschmaschine, die Trockneraktivitäten und die Warmwasserverbrauchsdaten, die in Kapitel 4.2 aufbereitet wurden, eingelesen und entsprechend bearbeitet. In Zeile 55-57 werden die Verbrauchsdaten der Waschmaschine berechnet, in Zeile 59-61 die des Wäschetrockners und in Zeile 64-68 die der Wärmepumpe. Somit werden Listen mit den Verbrauchsdaten der betrachteten Geräte pro Zeiteinheit erzeugt.

```
8  #Stromverbrauch der "smarten" Geräte in Wh pro Zeiteinheit
9  wmV = 245
10 trV = 500
11 boV = 200
14
15 #Anfangstemperatur der Wärmepumpe
16 anfangstemperatur = 55
17 temperaturNS = 55
18 temperaturdelta = -1
35
36 #berechnet den Stromverbrauch der smarten Geräte im nicht-smarten Modell und gibt die
       entsprechenden Listen zurück
37 class Nichtsmart:
38     def generate(wadaten, bodaten, trockneraktiv):
39         w = 0
52
53         for i in range(0,tagesAnzahl*zeitschrittepro Tag):
54             #Waschmaschine fängt direkt nach Auftragserteilung an zu waschen
55             if wadaten[w] == i:
56                 for j in range(0,4):
57                     waverbrauchNS[i+j] = wmV
58                 #Es wird überprüft, ob der Trockner aktiv ist oder aufgrund des
                       Wetters nicht angeschaltet wird. Wird der Trockner angestellt, lä
                       uft er direkt nach dem Waschvorgang.
59                 if trockneraktiv[w]==1:
60                     for k in range(0,3):
61                         trverbrauchNS[i+5+k] = trV
62                 if w < len(wadaten)-1:
63                     w += 1
64             #Temperatur wird neu berechnet. Sie wird immer versucht auf 55 Grad (
                   temperaturNS) zu halten, kann aber nicht mehr als 2 Grad (mit 200W)
                   erhöht werden.
65             temperatur = temperatur + temperaturdelta/10 - bodaten[i]/10
66             delta = min(200, temperaturNS*100 - temperatur*100)
67             boverbrauchNS.append(delta)
68             temperatur = temperatur + delta/100
```

Quellcode 10: Berechnung des konventionellen Modells (Auslesen.py).

4.5 Batterie

Der Strombedarf des betrachteten Haushalts wird zuerst durch den lokal erzeugten Strom gedeckt. Reicht dieser nicht aus, wird auf den gespeicherten Strom aus der Batterie zurückgegriffen. Besteht weiterhin ein Bedarf an Strom, wird dieser aus dem öffentlichen Stromnetz bezogen. Ist nach der Deckung durch den lokale erzeugten Strom jedoch ein Überschuss vorhanden, so wird dieser, wenn möglich, in der Batterie gespeichert und ansonsten in das öffentliche Stromnetz eingespeist.

Die Batterie wurde ebenso in Python implementiert. Dies ist in Quellcode 11 dargestellt. Zuerst wird in Quellcode 11 Zeile 135-142 aus dem Reststrom und dem Systemwirkungsgrad berechnet, wie viel überschüssiger lokal erzeugter Strom in die Batterie geladen werden könnte bzw. wie viel aus der Batterie geladen werden müsste, um den restlichen Strombedarf zu decken. Anschließend wird in Zeile 144-160 so viel Strom wie nötig und möglich aus bzw. in die Batterie geladen und der neue Batteriestand berechnet. Zudem wird der Rest ermittelt, der nicht mehr in die Batterie bzw. aus der Batterie geladen werden kann, weil diese bereits voll geladen bzw. entladen ist.

```
116 #Simuliert die Batterie
117 class Batterie:
128
129     #Bekommt den Reststrom (Differenz der PV-Produktion und des Gesamtverbrauchs, pos:
            Stromüberschuss, neg: Strommangel) übergeben und berechnet, wie viel in die
        Batterie bzw. aus der Batterie geladen wird.
130     #Es werden die neuen Batteriestände und der Rest, der nicht mehr in der Batterie
        gespeichert (positiv) bzw. nicht mehr aus der Batterie geladen werden konnte (
        negativ) übergeben.
131     def batteriestaende(self, restohnebatterie):
132         ladeplotrest = []
133         batteriestand = []
134
135         #Berechnet, wie viel überschüssige Solarenergie könnte in die Batterie geladen
                werden (positiv) bzw. wie viel aus der Batterie geladen werden müsste, um
                den restlichen Strombedarf zu decken (negativ)
136         ladeplot = []
137         rest = restohnebatterie
138         for i in range(0, len(rest)):
139             if rest[i] >= 0:
140                 ladeplot.append(rest[i] * self.systemwirkungsgrad)
141             else:
142                 ladeplot.append(rest[i] / self.systemwirkungsgrad)
143
144         #Der erste Batteriestand muss direkt gesetzt werden, um den neuen in der
            Schleife zu setzen.
145         batteriestand.append(self.startbatteriestand)
146         for i in range(0,len(ladeplot)):
147             if ladeplot[i] >= 0 and self.nutzbarekapazitaet >= batteriestand[i]+
                ladeplot[i]:
148                 batteriestand.append(batteriestand[i]+ladeplot[i])
```

```
149                 ladeplotrest.append(0)
150             elif ladeplot[i] >= 0 and self.nutzbarekapazitaet < batteriestand[i]+
                    ladeplot[i]:
151                 batteriestand.append(self.nutzbarekapazitaet)
152                 ladeplotrest.append((ladeplot[i]-(self.nutzbarekapazitaet-
                        batteriestand[i]))/self.systemwirkungsgrad)
153             elif ladeplot[i] < 0 and batteriestand[i] >= (-1)*ladeplot[i]:
154                 batteriestand.append(batteriestand[i]+ladeplot[i])
155                 ladeplotrest.append(0)
156             elif ladeplot[i] < 0 and batteriestand[i] < (-1)*ladeplot[i]:
157                 batteriestand.append(0)
158                 ladeplotrest.append((ladeplot[i]+batteriestand[i])*self.
                        systemwirkungsgrad)
159         #Der Startbatteriestand muss wieder gelöscht werden.
160         del batteriestand[0]
```

Quellcode 11: Implementierung der Batterie (Auslesen.py).

4.6 Analyse

Um zu analysieren, welche Vorteile die Maximierung des Eigenverbrauchs für die verschiedenen Haushalte und Intervallgruppen liefert, wurde der Eigenverbrauch, die Effizienz der einzelnen Geräte und die Kosten untersucht. Auch dies wurde in Python realisiert.
Der Eigenverbrauch stellt den Anteil des lokal erzeugten Stroms dar, der für den eigenen Strombedarf genutzt wird, und wird daher wie folgt berechnet:

$$Eigenverbrauch = \frac{Solarnutzung}{Jahresproduktion}.$$

Die Solarnutzung besteht aus dem lokal erzeugten Strom, der nicht in das öffentliche Stromnetz eingespeist, sondern für den Eigenverbrauch genutzt wird.

Es wird davon ausgegangen, dass der lokal produzierte Strom bzw. der gespeicherte Strom aus der Batterie, zuerst den Strombedarf der konventionellen Geräte deckt und der Rest anschließend auf die aktiven smarten Geräte gleichermaßen aufgeteilt wird. Um die Effizienz der einzelnen Geräte zu erhalten, wird somit berechnet, inwieweit deren Strombedarf durch die PV-Anlage und die Batterie gedeckt wird.

Die Kosten des Stromein- und Stromverkaufs wurden unter Berücksichtigung des durchschnittlichen Strompreises von 28,81 Cent/kWh im Jahr 2015 für deutsche Privathaushalte [6] und einer Einspeisevergütung von 12,5 Cent/kWh [5] berechnet. Dabei wurde davon ausgegangen, dass die betrachteten Häuser die PV-Anlage im November 2015 zum ersten Mal in Betrieb genommen haben und sich diese Anlage auf dem Dach des Hauses befindet.

5 Ergebnisse

Zur genauen Analyse der Ergebnisse aus Kapitel 4.6, wird zuerst der Eigenverbrauch für jeden Haushalt in Hinblick auf die verschiedenen Intervalle und Modelle untersucht. Dies zeigt, in welchem Maße der lokal produzierte Strom genutzt wird, sagt allerdings nichts über den Einfluss der PV-Anlage aus. So erhöht eine größere Anlage beispielsweise den Deckungsgrad des Strombedarfs durch lokal erzeugten Strom, verringert jedoch den Eigenverbrauch [3, S.26]. Um dennoch den Einfluss auf die Effizienz der PV-Anlage beurteilen zu können, werden die entstandenen Stromkosten, sowie die Effizienz der einzelnen Geräte betrachtet, d. h. die Deckung des Strombedarfs durch den lokal erzeugten Strom.
Zudem wird in den smarten Modellen zusätzlich ein Modell ohne smarte Warmwasser-Wärmepumpe in der ersten Intervallgruppe betrachtet, um die Auswirkungen dieser Intervalllängen auf die Ergebnisse im Vergleich zu den konventionellen Modellen beurteilen zu können.
Folglich gibt es pro Haushalt je 6 Modelle, die untersucht werden. Dabei wird jedes Modell mit und ohne Batterie betrachtet (siehe Tabelle 7).

	Wärmepumpe	**Intervalllänge der Waschmaschine** (h)	**Intervalllänge des Trockners** (h)	**Batterie**
1. Modell	×	2	1,5	× ✓
2. Modell	×	3	2	× ✓
3. Modell	✓	3	2	× ✓
4. Modell	✓	4	3	× ✓
5. Modell	✓	6	4,5	× ✓
6. Modell	✓	8	6	× ✓

Tabelle 7: Betrachtete Modelle

Bei der Wärmepumpe wird lediglich angegeben, ob diese smart ist oder nicht. Bei

der Waschmaschine und dem Trockner werden die jeweiligen Intervalllängen benannt, wobei die Intervalllänge von 2 Stunden bei der Waschmaschine und von 1,5 Stunden bei dem Trockner bedeutet, dass diese nicht smart sind. Das 1. Modell ist also das konventionelle Modell.
Abschließend werden die Haushalte untereinander verglichen, um zu beurteilen, für welchen Haushaltstyp die Maximierung des Eigenverbrauchs am lohnenswertesten ist.

5.1 Haushalt der Familie

In Abbildung 12 ist der Eigenverbrauch der Familie in den betrachteten Modellen dargestellt. Auf der Abszisse sind die verschiedenen Modelle abgebildet, wobei die dunkelblauen Säulen das jeweilige Modell ohne Batterie darstellt und die hellblauen Säulen das jeweilige Modell mit Batterie. Der Eigenverbrauch in Prozent ist auf der Ordinate zu sehen. Die Prozente in den Kreisen auf den Säulen geben jeweils die Steigerung zum vorherigen Modell an.

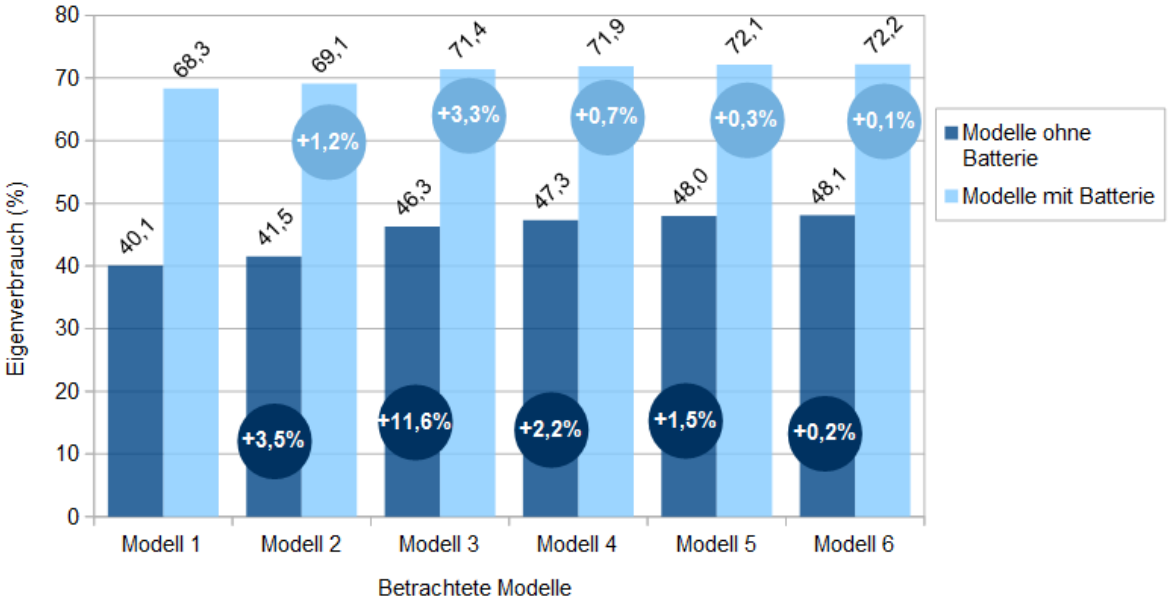


Abbildung 12: Eigenverbrauch des Haushalts der Familie in den betrachteten Modellen

Wird das konventionelle Modell mit dem 2. Modell verglichen, so ist ein Anstieg des Eigenverbrauchs zu verzeichnen. Das heißt bereits eine smarte Waschmaschine und ein smarter Trockner steigern den Eigenverbrauch der Familie. Trotz der kleinen Intervalllängen von 3 bzw. 2 Stunden der Waschmaschine bzw. des Trockners kann das Verbrauchsmuster dieser Geräte durch den optimalen Schedule so verschoben werden, dass der lokal erzeugte Strom besser genutzt werden kann. In den Modellen ohne Bat-

terie ist eine Steigerung von 3,5 % und in denen mit Batterie von 1,2 % zu verzeichnen. Eine größere Steigerung von 11,6 % bzw. 3,3 % ist hin zum 3. Modell zu erkennen. Die smarte Wärmepumpe unterstützt die Maximierung des Eigenverbrauchs erheblich. Der Überschuss an lokal produziertem Strom kann indirekt in dieser gespeichert werden, indem das Wasser weiter erhitzt wird. Dies muss somit weniger zu Zeiten des lokalen Strommangels geschehen.
Die Vergrößerung der Intervalllängen im 4. Modell sorgt wiederum für eine kleine Steigerung von etwa 2,2 % bzw. 0,7 % im Vergleich zum 3. Modell. Aufgrund dieser Ausweitung auf 4 Stunden bei der Waschmaschine und auf 3 Stunden bei dem Trockner kann der Schedule dieser Geräte nochmals verbessert werden. In den Modellen 5 und 6 ist kaum noch eine Erhöhung des Eigenverbrauchs zu erkennen. Dies lässt darauf schließen, dass ab einer gewissen Intervalllänge bei einer Vergrößerung dieser keine Steigerung mehr stattfindet. Dies liegt vermutlich daran, dass ab einer bestimmten Länge der Zeitraum eines lokal produzierten Stromüberschuss bereits im Intervall liegt. Es wird somit immer unwahrscheinlicher, dass eine weitere Verlängerung des Intervalls einen Zeitraum größeren Stromüberschusses aufweist.
Bei dem Haushalt der Familie lässt sich also sagen, dass eine Intervalllänge der Waschmaschine von 4 Stunden und des Trockners von 3 Stunden in den Modellen ohne Batterie bereits eine gute Steigerung des Eigenverbrauchs bewirken und weitere Verlängerungen der Intervalle keinen großen Einfluss haben. In den Modellen mit Batterie gibt es bereits ab einer Intervalllänge von 3 bzw. 2 Stunden keine signifikante Steigerung mehr.
Es ist zudem deutlich zu erkennen, dass eine Batterie als Energiespeicher eine sehr gute Möglichkeit ist, den Eigenverbrauch zu steigern. Selbst im konventionellen Modell lässt sich durch diese eine Erhöhung von etwa 40 % auf rund 68 % verzeichnen. Durch das Speichern des überschüssigen lokal produzierten Stroms hat die Erhöhung der Intervalllängen einen geringeren Einfluss auf den Eigenverbrauch. Liegt das Intervall nicht in dem Zeitraum eines lokal erzeugten Stromüberschuss, so kann dieser in der Batterie gespeichert und später den Geräten zur Verfügung gestellt werden.

In Abbildung 13 sind die jährlichen Stromkosten des Haushalts der Familie für die verschiedenen Modellen dargestellt. Auf der Abszisse sind wieder die betrachteten Modelle ohne Batterie (dunkelblau) und mit Batterie (hellblau) dargestellt. Die Kosten sind in Euro auf der Ordinate aufgetragen. In den Kreisen ist der prozentuale Unterschied zum jeweils vorherigem Modell dargestellt.
Es ist eine Einsparung vom konventionellem zum zweiten Modell zu erkennen. Durch die smarte Waschmaschine und den smarten Trockner können in dem Modell ohne

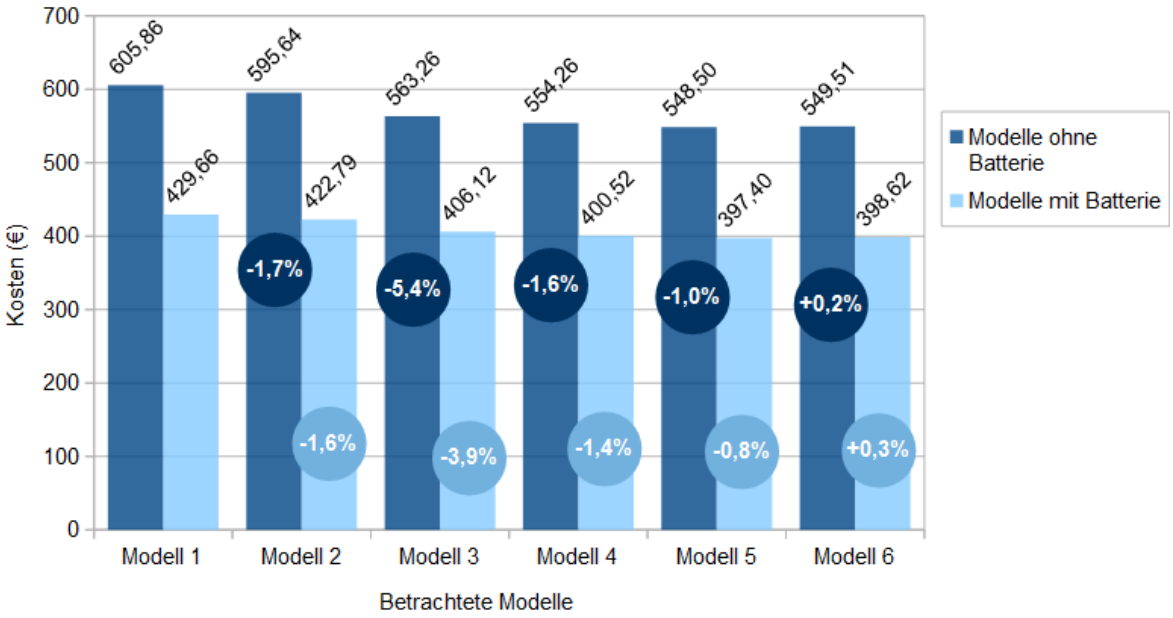


Abbildung 13: Jährliche Stromkosten des Haushalts der Familie in den betrachteten Modellen

Batterie 1,7 % gespart werden. Die smarte Wärmepumpe erzielt jedoch eine größere Ersparnis. In Modell 3 ohne Batterie belaufen sich die jährlichen Stromkosten auf rund 563 €. Dies ist eine Ersparnis von 5,4 % im Vergleich zum zweiten Modell. Im vierten Modell werden ohne Batterie nochmals 1,6 % eingespart und mit Batterie 1,4 %. Es ist somit zu erkennen, dass die Verlängerung der Intervalle der Waschmaschine und des Trockners durchaus Einfluss auf die Stromkosten hat. Jedoch wird im 5. und 6. Modell wieder deutlich, dass der Einsparungsanteil immer weiter abnimmt, je größer die Intervalle werden. Im 5. Modell gibt es nur noch eine Ersparnis von 1,0 % bzw. 0,8 %.
Auch bei den Stromkosten erzielt die Maximierung des Eigenverbrauchs bis zu einer Intervalllänge von 4 bzw. 3 Stunden eine sichtbare Ersparnis. Eine weitere Verlängerung der Intervalle scheint jedoch keine signifikanten Einsparungen hervorzurufen. Ebenso ist deutlich zu erkennen, dass eine Batterie von großem Vorteil ist. So können im konventionellen Modell bereits gut 176 € eingespart werden. Aber auch durch die smarten Geräte sind Einsparungen möglich. So können im 4. Modell im Vergleich zum konventionellen Modell fast 7 % in den Batteriemodellen eingespart werden.

Da die smarten Geräte bisher zusammenfassend betrachtet wurden, wird im Folgenden auf die Effizienz der einzelnen Haushaltsgeräte eingegangen. Dazu ist in Abbildung 14 die prozentuale Deckung des Strombedarfs der einzelnen Geräte durch den lokal pro-

duzierten Strom für die Modelle ohne Batterie dargestellt. Zudem ist die prozentuale Deckung des Strombedarfs der Geräte zusammengefasst zu sehen. Auf der Abszisse sind die Geräte in Gruppen dargestellt. Die verschiedenen Farben stellen die jeweiligen Modelle dar. Auf der Ordinate ist die Effizienz in % aufgetragen. Die Werte in den Kreisen geben die Steigerung zwischen den Modellen für jedes Gerät an.

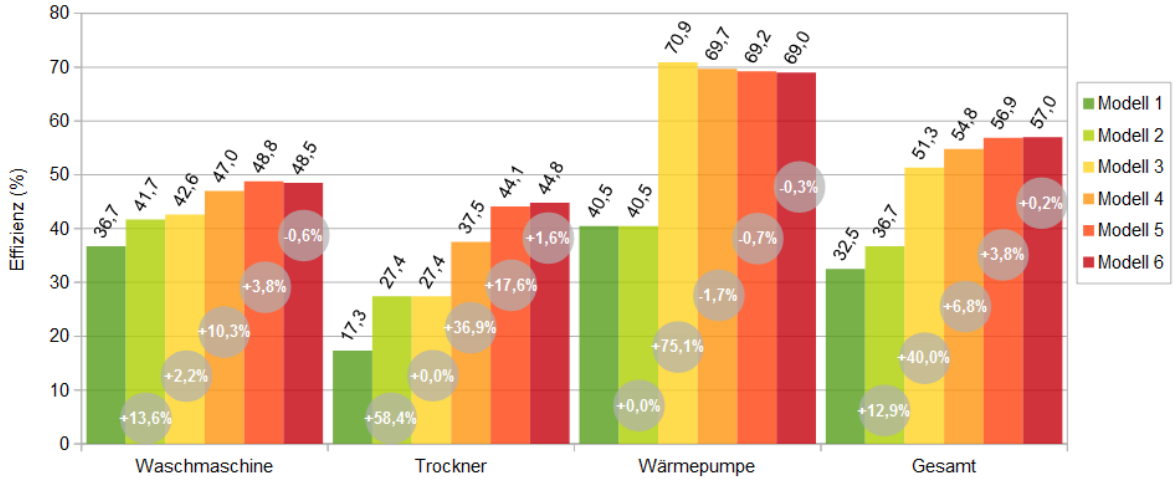


Abbildung 14: Effizienz der betrachteten Haushaltsgeräte in den Modellen ohne Batterie

Die Deckung des Strombedarfs der Waschmaschine und des Trockners durch die lokale Stromproduktion steigt jeweils vom konventionellen Modell zum 2. Modell. Die Steigerung der Deckung des Trockners ist jedoch deutlich höher. Dies liegt vermutlich daran, dass durch die Verlängerung des Intervalls die Endzeitpunkte des Trockners des öfteren auf den nächsten Morgen verschoben werden. Dies geschieht, wenn der Endzeitpunkt eigentlich auf eine Zeit nach Mitternacht fallen würde. Zudem fällt der Endzeitpunkt des Trockners auf den nächsten Morgen, wenn der Endzeitpunkt der Waschmaschine ebenfalls auf den nächsten Morgen verschoben wird. Daher ist die Steigerung der Deckung des Trockners so enorm mit 58,4 %. Da die Wärmepumpe in beiden Modellen nicht smart ist, bleibt ihre Effizienz in etwa gleich. Somit steigt die Gesamtdeckung der betrachteten Geräte von 32,5 % auf 36,7 %.

Die Hinzunahme einer smarten Wärmepumpe im dritten Modell ist von großem Vorteil für die Effizienz der Geräte. So steigt die Deckung der betrachteten Geräte um 40,0 %. Betrachtet man die Deckung der einzelnen Geräte, so ist eine deutliche Steigerung von 75,1 % bei der Wärmepumpe zu verzeichnen. Durch die Verschiebung des Schedules dieser wird, ändert sich möglicherweise auch der Schedule der Waschmaschine und des Trockners leicht. Zudem wird dadurch der lokal produzierte Strom anders verteilt. Somit steigt die Effizienz der Waschmaschine um 2,2 %.

Um die Auswirkungen der Intervalllängen zu beurteilen, sind vor allem die weiteren Modelle interessant. Da sich in diesen die Eigenschaften der Wärmepumpe nicht ändern, bleibt die Deckung dieser in den Modellen 3-6 in etwa gleich. Die leichte Reduktion der Deckung ist dem veränderten Schedule und der Umverteilung des lokal produzierten Stroms zu verschulden. Bei der Deckung des Strombedarfs der Waschmaschine und des Trockners ist jedoch ein deutlicher Anstieg zu verzeichnen. So steigt die Deckung der Waschmaschine im 4. Modell um 10,3 % und die des Trockners sogar um 36,9 %. Dies ergibt eine Steigerung der Gesamtdeckung von 51,3 % auf 54,8 %. Es sind also im 4. Modell deutliche Steigerungen zu sehen.
Im 5. Modell hingegen sind diese eher gering. So steigt die Effizienz der Waschmaschine nur noch um 3,8 %. Die des Trockners erhöht sich zwar um 17,6 %, jedoch steigt die Gesamtdeckung nur um etwa 3,8 %. Die Gesamtdeckung im 6. Modell verzeichnet kaum einen Anstieg.
Es lässt sich somit für die Effizienz der betrachteten Geräte festhalten, dass alle drei smarten Geräte einen großen Einfluss auf die Deckung haben. Auf die Intervalllängen bezogen ist vor allem bei der Waschmaschine die Verlängerung von 3 auf 4 Stunden und beim Trockner die von 2 auf 3 Stunden ausschlaggebend.

In Abbildung 15 ist die prozentuale Deckung des Strombedarfs der Geräte durch den lokal erzeugten Strom inklusive des in der Batterie gespeicherten Stroms dargestellt.

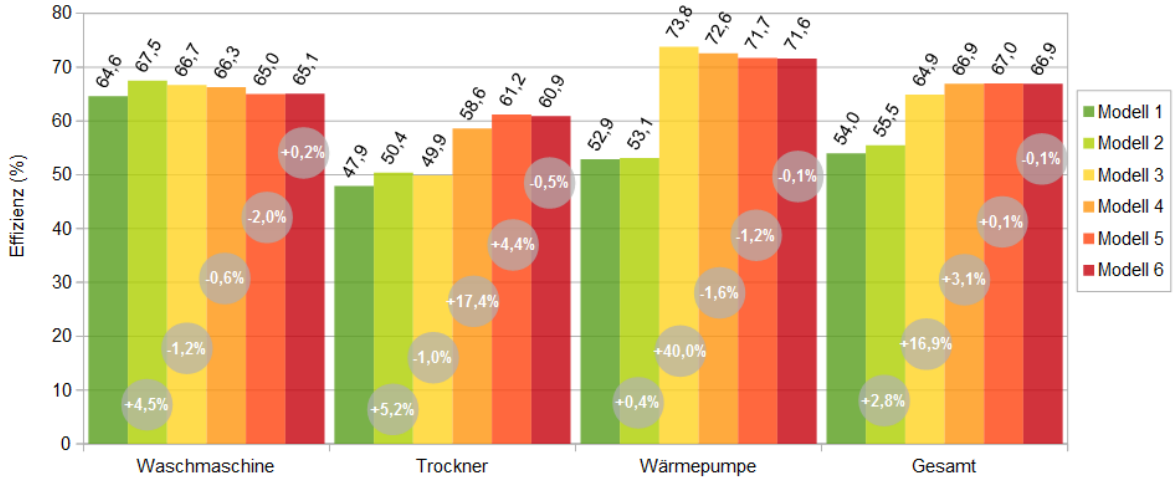


Abbildung 15: Effizienz der betrachteten Haushaltsgeräte in den Modellen mit Batterie

Es ist zu erkennen, dass die Effizienz aller Geräte sichtlich höher ist, als in den Modellen ohne Batterie. Deutliche Steigerungen sind insgesamt zwischen dem konventionellem und smarten Modell zu sehen, jedoch nicht durch die Verlängerung der Intervalle der

Waschmaschine bzw. des Trockners. So steigt die Effizienz der Waschmaschine vom 1. zum 3. Modell um etwa 3,3 %, die des Trockners um 4,2 % und die der Wärmepumpe sogar um 39,5 %. Somit ergibt sich eine Gesamtsteigerung von über 20 %.
Vom 3. zum 4. Modell kann der Schedule durch die Vergrößerung der Intervalle zwar so verschoben werden, dass die Deckung des Trockners deutlich steigt. Die Deckung der Waschmaschine und der Wärmepumpe sinkt jedoch leicht. Somit erhöht sich die Gesamtdeckung nur um 3,1 %.
Bei den Batteriemodellen wird die Effizienz der Geräte folglich hauptsächlich dadurch beeinflusst, ob diese smart oder nicht smart sind. Eine Vergrößerung der Intervalle hat keinen großen Einfluss mehr.

Es lässt sich somit für den Haushalt der Familie zusammenfassen, dass gerade in den Modellen ohne Batterie die smarten Geräte von Vorteil sind. Es ist auch eine deutliche Verbesserung durch die Vergrößerung der Intervalle auf 4 Stunden bei der Waschmaschine und auf 3 Stunden bei dem Trockner zu sehen. Eine weitere Vergrößerung hat jedoch keinen signifikanten Einfluss. In den Batteriemodellen ist eine ähnliche Tendenz zu verzeichnen, jedoch mit geringeren Steigerungen. Dies hat zur Folge, dass eine Vergrößerung der Intervalllängen keinen signifikanten Einfluss hat. Im Vergleich zu den Modellen ohne Batterie liefern die Batteriemodelle in allen Bereichen bessere Werte. Jedoch haben eine Batterie und die smarten Geräte auch einen gewissen Anschaffungswert. Auf diesen wird in dieser Arbeit jedoch nicht näher eingegangen.

5.2 Haushalt des Rentnerpaares

In Abbildung 16 ist der Eigenverbrauch des Rentnerpaares in den 6 Modellen ohne (dunkelblaue Säulen) und mit Batterie (hellblaue Säulen) dargestellt. Auf der Abszisse sind die verschiedenen Modelle und auf der Ordinate der Eigenverbrauch in % abgebildet. Die Werte in den Kreisen geben die Steigerung zwischen den Modellen an.

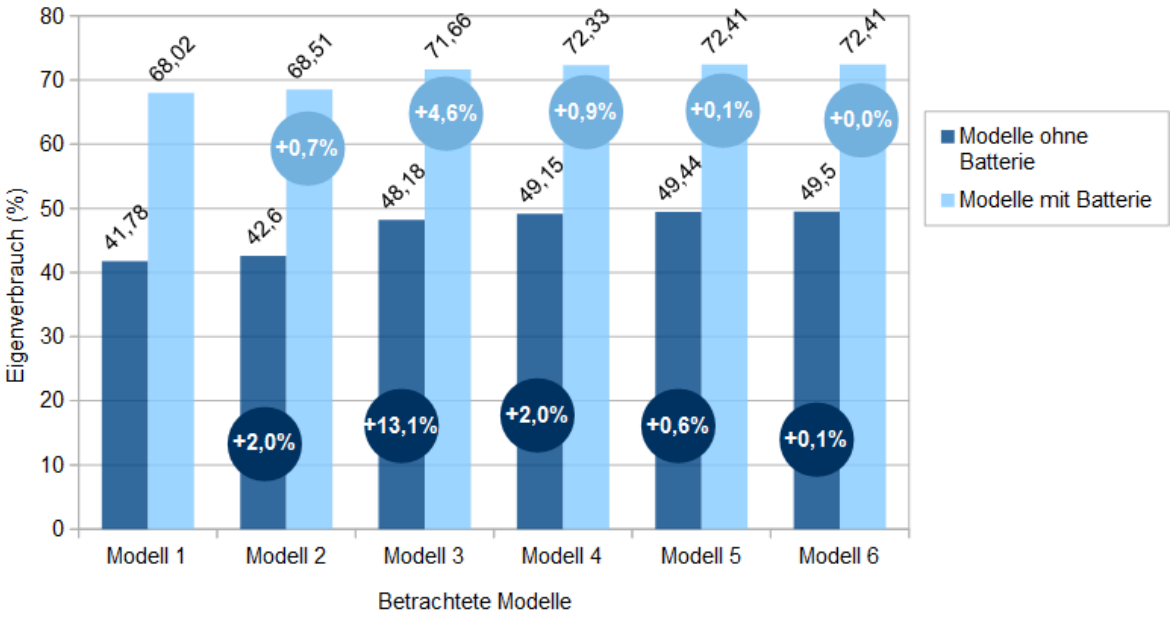


Abbildung 16: Effizienz der betrachteten Haushaltsgeräte in den Modellen mit Batterie

Im Diagramm ist zu erkennen, dass im 2. Modell der Schedule der Waschmaschine und des Trockners erfolgreich verschoben wird, sodass der Eigenverbrauch im Vergleich zum konventionellen Modell um 1,9 % steigt. In den Modellen mit Batterie hingegen ist nur eine Erhöhung um 0,7 % zu verzeichnen.
Die smarte Wärmepumpe hat einen großen Einfluss auf den Eigenverbrauch. So ist ein deutlicher Anstieg in beiden Modellen von 13,1 % bzw. 4,7 % im 3. Modell zu sehen. Eine Vergrößerung der Intervalllängen der Waschmaschine und des Trockners hat hauptsächlich Auswirkungen auf die Modelle ohne Batterie. So steigt der Eigenverbrauch von 48,2 % auf 49,2 %. In den Modellen mit Batterie ist jedoch nur ein Anstieg von 0,8 % zu verzeichnen. Eine weitere Verlängerung der Intervalle ruft in beiden Modellen lediglich eine geringe Steigerung hervor.
Vergleicht man die Modelle ohne Batterie mit denen mit Batterie, so ist auch hier eine deutliche Steigerung zu erkennen. Durch das Speichern des lokal erzeugten Stroms können beispielsweise im 3. Modell fast 50 % eingespart werden.

Für den Eigenverbrauch des Rentnerpaares lässt sich festhalten, dass gerade die smarte Wärmepumpe und eine Batterie von Vorteil sind. In den Modellen ohne Batterie liefert zudem die Verlängerung der Intervalle der Waschmaschine auf 4 Stunden und des Trockners auf 3 Stunden eine Verbesserung.

Abbildung 17 liefert die jährlichen Stromkosten des Rentnerpaares. Auf der Abszisse sind die verschiedenen Modelle und auf der Ordinate die jährlichen Stromkosten in € dargestellt. Die Werte in den Kreisen geben die prozentuale Ersparnis zwischen den sechs Modellen an.

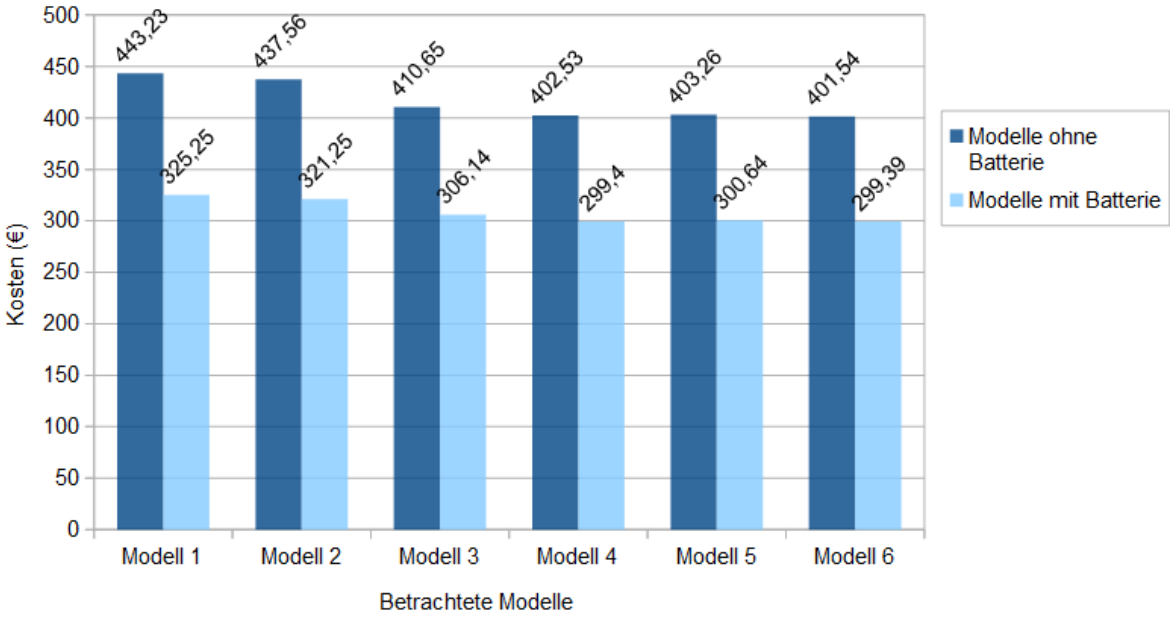


Abbildung 17: Effizienz der betrachteten Haushaltsgeräte in den Modellen mit Batterie

Im zweiten Modell kann durch den optimierten Schedule der smarten Waschmaschine und des smarten Trockners in beiden Modellen fast 1,0 % eingespart werden. Somit wird deutlich, dass schon eine kleine Ausweitung der Intervalle dieser Geräte zur Minimierung der Stromkosten beiträgt. Die Wärmepumpe hat jedoch einen größeren Einfluss. So sinken die Stromkosten im Modell mit Batterie um 5,0 % und im Modell ohne Batterie sogar um 6,2 %. Durch eine Vergrößerung der Intervalle sinken die Stromkosten im 4. Modell auf 401,58 € bzw. 298,52 €. Eine weitere Vergrößerung lässt die Stromkosten jedoch nur noch leicht sinken.
Eine Batterie als Energiespeicher ist in allen sechs Modellen von Vorteil. So liegt die Ersparnis in allen Modellen bei etwa 26 %.
Insgesamt lässt sich sagen, dass die prozentuale Ersparnis in den Batteriemodellen und

den Modellen ohne Batterie ähnlich sind. Vor allem durch die Wärmepumpe können Kosten eingespart werden.

In Abbildung 18 ist die Effizienz der einzelnen Geräte durch die prozentuale Deckung des Stromverbrauchs dieser durch den lokal erzeugten Strom in den sechs Modellen ohne Batterie dargestellt. Auf der Abszisse sind die verschiedenen Geräte, sowie diese zusammengefasst aufgetragen. Die Effizienz in Prozenz ist auf der Ordinate zu sehen. Die Werte in den Kreisen stellen für jedes Gerät die prozentuale Steigerung zwischen den Modellen dar.

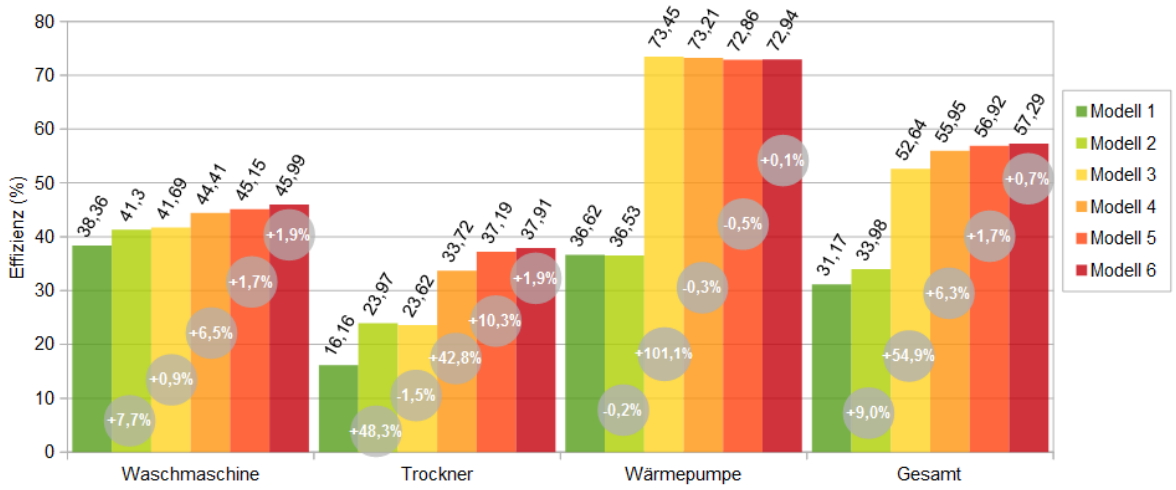


Abbildung 18: Effizienz der betrachteten Haushaltsgeräte in den Modellen mit Batterie

Im Gegensatz zum konventionellen Modell ist die Effizienz der Waschmaschine und des Trockners im zweiten Modell größer. Vor allem der Schedule des Trockners wird erfolgreich vorschoben. So steigt die Effizienz bei diesem um 48,1 %. Durch diese Verschiebung der Schedules wird der lokal produzierte Strom anders auf die Geräte verteilt. Dies erklärt die minimale Reduktion der Effizienz der Wärmepumpe. Trotzdem entsteht eine Gesamtsteigerung von 9,0 %.
Die größte Erhöhung findet im vierten Modell statt, in dem die Effizienz der smarten Geräte auf 54,7 % steigt. Dies wird von der smarten Wärmepumpe hervorgerufen, deren Effizienz sich im Vergleich zum zweiten Modell mehr als verdoppelt. Die leichten Schwankungen der Waschmaschine und des Trockners sind der oben genannten Umverteilung des lokal produzierten Stroms zu verschulden.
In den weiteren Modellen bleibt die Effizienz der Wärmepumpe in etwa gleich, da sich die Eigenschaften dieser in den Modellen nicht mehr ändert. Die Deckung der Waschmaschine und des Trockners steigt jedoch weiter, da die Verlängerung der Intervalle zu

einem optimierten Schedule führt. So steigt die Gesamtdeckung im vierten Modell auf 56 %. Eine weitere Vergrößerung der Intervalllängen führt bei der Deckung des Trockners zwar zu einer deutlichen Steigerung von 10 %, die Gesamteffizienz steigt jedoch nur um 1,6 %, da der lokal erzeugte Strom hauptsächlich nur anders verteilt wurde. Im sechsten Modell ist Ähnliches zu beobachten.
Es lässt sich somit sagen, dass bis zum viertel Modell deutliche Steigerungen zu erkennen sind. Eine weitere Verlängerung der Intervalle hat jedoch keinen großen Einfluss mehr.

Die Effizienz der einzelnen Geräte in den sechs Modellen mit Batterie ist in Abbildung 19 abgebildet. So sind auf der Abszisse wieder die verschiedenen Geräte, sowie diese zusammengefasst dargestellt und auf der Ordinate die Effizienz in Prozent. Auch in dieser Abbildung stellen die Werte in den Kreisen für jedes Gerät die prozentuale Steigerung zwischen den Modellen dar.

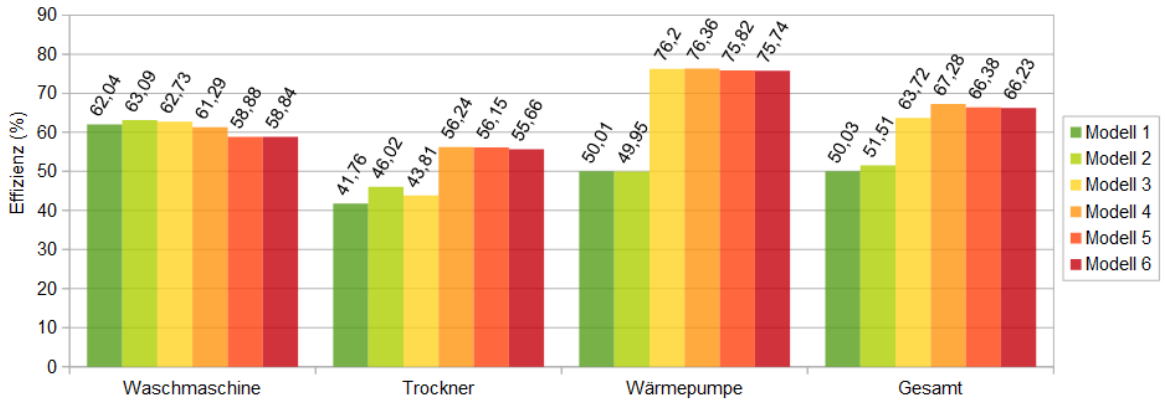


Abbildung 19: Effizienz der betrachteten Haushaltsgeräte in den Modellen mit Batterie

Auch in den Modellen mit Batterie ist eine Steigerung der Effizienz zwischen dem konventionellen und zweiten Modell zu verzeichnen. Jedoch ist diese mit 2,9 % deutlich geringer als die Effizienz in den Modellen ohne Batterie. Die smarte Wärmepumpe hingegen steigert ihre Effizienz um über 50 % und lässt somit die Gesamtdeckung im dritten Modell auf 65,3 % ansteigen. Aufgrund des optimierten Schedules der Wärmepumpe, ändern sich auch die anderen Schedules und die Verteilung des lokal erzeugten Stroms. Dies hat zur Folge, dass die Deckung der Waschmaschine und des Trockners jeweils um über 4 % sinken.
Eine Verlängerung der Intervalle bewirkt zwar eine Steigerung der Effizienz des Trockners, jedoch steigt die Gesamteffizienz lediglich um 3,7 %. Durch die Speicherung des lokal erzeugten Stroms in der Batterie kann dieser zu einem Zeitpunkt verwendet wer-

den, an dem Bedarf besteht. Somit hat eine Vergrößerung der Intervalle keinen großen Einfluss.
In den Modellen fünf und sechs sinkt die Gesamteffizienz sogar minimal. Der Schedule der Waschmaschine und des Trockners hat sich so verschoben, dass der gespeicherte Strom mehr für die konventionellen Geräte verwendet wird und somit weniger für die smarten Geräte vorhanden ist.
Insgesamt ist auch zu erkennen, dass die Deckungen in den Modellen ohne Batterie in etwa doppelt so hoch sind wie die in den Modellen mit Batterie. Die smarten Geräte haben folglich deutlich mehr Einfluss auf die Effizienz, wenn keine Batterie als Energiespeicher genutzt wird. Jedoch lässt gerade eine Wärmepumpe die Effizienz steigen. Die Verlängerung der Intervalle auf vier Stunden bei der Waschmaschine und auf drei Stunden beim Trockner bringen allerdings nur einen kleinen Vorteil.

Abschließend lässt sich für den Haushalt des Rentnerpaares festhalten, dass gerade die smarte Wärmepumpe und die Batterie von Vorteil sind. Es ist somit lohnend Geräte zu verwenden, die in gewisser Weise Energie speichern können. Aber vor allem in den Modellen ohne Batterie konnte durch die Verlängerung der Intervalle der Waschmaschine auf vier Stunden und des Trockners auf drei Stunden eine leichte Verbesserung erzielt werden. In den Modellen mit Batterie hingegen konnten die smarte Waschmaschine und der smarte Trockner keinen großen Einfluss nehmen. Der gespeicherte Strom konnte zu Bedarfszeiten genutzt werden, um den Strombedarf der Waschmaschine und des Trockners zu decken. Somit hatte der Schedule dieser Geräte keine große Wirkung.

6 Fazit

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Literatur

[1] Bundesministerium für Wirtschaft und Energie, "BMWi - Erneuerbare Energien auf einen Blick," (Zugriff: 07.11.2016). [Online]. Available: https://www.bmwi.de/DE/Themen/Energie/Erneuerbare-Energien/erneuerbare-energien-auf-einen-blick.html

[2] Presse- und Informationsamt der Bundesregierung, "Erneuerbare Energien: Ein neues Zeitalter hat begonnen," (Zugriff:10.11.2016). [Online]. Available: https://www.bundesregierung.de/Webs/Breg/DE/Themen/Energiewende/EnergieErzeugen/ErneuerbareEnergien-Zeitalter/_node.html

[3] H. Wirth, "Aktuelle Fakten zur Photovoltaik in Deutschland," Ph.D. dissertation, Frauenhofer ISE, Oktober, 2016.

[4] Bundesnetzagentur, "Archivierte Datenmeldungen," 2016, (Zugriff: 10.11.2016). [Online]. Available: http://www.bundesnetzagentur.de/cln_1422/DE/Sachgebiete/ElektrizitaetundGas/Unternehmen_Institutionen/ErneuerbareEnergien/Photovoltaik/ArchivDatenMeldgn/ArchivDatenMeldgn_node.html

[5] "Datenmeldungen und EEG-Vergütungssätze für Photovoltaikanlagen," *Bundesnetzagentur*, 31.10.2016, (Zugriff: 03.11.2016). [Online]. Available: http://www.bundesnetzagentur.de/DE/Sachgebiete/ElektrizitaetundGas/Unternehmen_Institutionen/ErneuerbareEnergien/Photovoltaik/DatenMeldgn_EEG-VergSaetze/DatenMeldgn_EEG-VergSaetze_node.html

[6] "BDEW zum Strompreis der Haushalte," *BDEW Bundesverband der Energie- und Wasserwirtscht e.V.*, 2015, (Zugriff: 03.11.2016). [Online]. Available: https://www.bdew.de/internet.nsf/id/9D1CF269C1282487C1257E22002BC8DD/$file/150409%20BDEW%20zum%20Strompreis%20der%20Haushalte%20Anhang.pdf

[7] O. Bendel, *Gabler Wirtschaftslexikon, Stichwort: Smart Home.* Springer Gabler Verlag, (Zugriff: 10.11.2016). [Online]. Available: http://wirtschaftslexikon.gabler.de/Archiv/-2046533094/smart-home-v2.html

[8] Y.-S. Son, T. Pulkkinen, K.-D. Moon, and C. Kim, "Home energy management system based on power line communication," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1380–1386, 2010.

[9] G. Behrmann, "UPPAAL CORA," 04.02.2014, (Zugriff: 02.11.2016). [Online]. Available: http://people.cs.aau.dk/~adavid/cora/index.html

[10] G. Hoogsteen, "GENETX/alpg - Artificial Load Profile Generator for DSM," (Zugriff: 15.11.2016). [Online]. Available: https://github.com/GENETX/alpg

[11] F. Stein, "Synthesis of smart appliances in smart homes with batteries to maximize self-use," Bachelor Thesis, Westfälische Wilhelms-Universität Münster, Münster, Deutschland, August, 2016.

[12] G. Hoogsteen, A. Molderink, J. L. Hurink, and G. J. Smit, "Generation of flexible domestic load profiles to evaluate Demand Side Management approaches," in *2016 IEEE International Energy Conference (ENERGYCON)*, pp. 1–6.

[13] M. Jongerden, J. Hüls, A. Remke, and B. Haverkort, "Does Your Domestic Photovoltaic Energy System Survive Grid Outages?" *Energies*, vol. 9, no. 9, p. 736, 2016.

[14] P. Jochem, *Gabler Wirtschaftslexikon, Stichwort: Demand Side Management (DSM).* Springer Gabler Verlag, (Zugriff: 16.11.2016). [Online]. Available: http://wirtschaftslexikon.gabler.de/Archiv/576005948/demand-side-management-dsm-v4.html

[15] K. G. Larsen, P. Petterson, and W. Yi, "UPPAAL in a Nutshell," *Springer Internation Journal of Software Tools for Technology Transfer*, 1997, (Zugriff: 27.10.2016). [Online]. Available: https://www.it.uu.se/research/group/darts/papers/texts/lpw-sttt97.pdf

[16] G. Behrmann, "UPPAAL CORA - Introduction," 04.02.2014, (Zugriff: 03.11.2016). [Online]. Available: http://people.cs.aau.dk/~adavid/cora/introduction.html

[17] A. David, T. Amnell, and M. Stigge, "UPPAAL 4.0: Small Tutorial," in *Optimization Principles*, N. S. Rau, Ed. IEEE, 2003.

[18] "UP4ALL Inc - GUI Reference," 2012, (Zugriff: 16.11.2016). [Online]. Available: http://www.uppaal.com/index.php?sida=216&rubrik=101

[19] G. Behrmann, "UPPAAL CORA - New Command Line Options," 04.02.2014, (Zugriff: 24.10.2016). [Online]. Available: http://people.cs.aau.dk/~adavid/cora/options.html

[20] G. Behrmann, A. Fehnker, T. S. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. W. Vaandrager, "Minimum-Cost Reachability for Priced Timed Automata," *BRICS Report Series*, no. 3, 2001.

[21] B. Stoimenova, "Echtzeitsysteme mit kontinuierlicher Zeit," Seminar, TU München, München, (Zugriff: 18.10.16). [Online]. Available: http://wind.informatik.tu-muenchen.de/seminare/model-checking/SS2002/stoimenova.pdf

[22] D. L. Dill, "Timing assumptions and verification of finite-state concurrent systems," in *Automatic Verification Methods for Finite State Systems*, ser. Lecture Notes in Computer Science, J. Sifakis, Ed. Berlin, Heidelberg: Springer-Verlag, 1990, vol. 407, pp. 197–212.

[23] G. Behrmann, K. G. Larsen, and J. I. Rasmussen, "Priced Timed Automata: Algorithms and Applications," in *Formal methods for components and objects*, ser. Lecture Notes in Computer Science, F. S. de Boer, Ed. Berlin, Heidelberg: Springer-Verlag, 2005, vol. 3657, pp. 162–182, (Zugriff: 01.11.2016). [Online]. Available: http://www.artes.uu.se/events/summer06/p2.pdf

[24] R. Abboud, S. Averkamp, A.-L. Linnemann, and F. Stein, "Maximierung der effizienten Eigennutzung der Stromproduktion in Smart Homes," Ph.D. dissertation, Westfälische Wilhelms-Universität Münster, Münster, Deutschland, Februar, 2016.

[25] G. Behrmann, A. David, and K. G. Larsen, "A Tutorial on UPPAAL 4.0," Ph.D. dissertation, Aalborg University, Aalborg, 2006, (Zugriff: 23.10.16). [Online]. Available: http://www.uppaal.com/admin/anvandarfiler/filer/uppaal-tutorial.pdf

[26] "UP4ALL Inc - Language Reference," 2012, (Zugriff: 24.10.2016). [Online]. Available: http://www.uppaal.com/index.php?sida=217&rubrik=101

[27] "KNMI - Uurgegevens van het weer in Nederland," (Zugriff: 25.10.2016). [Online]. Available: http://projects.knmi.nl/klimatologie/uurgegevens/selectie.cgi

[28] J. von Appen, B. Brugenmeister, Braun, and Martin, "Optimale Dimensionierung von PV-Speicher-Systemen unter Unsicherheit," in *30. Symposium Photovoltaische Solarenergie*, März, 2015.

[29] "kWp - Kilowatt-Peak | Spitzenleistung," (Zugriff: 27.10.2016). [Online]. Available: http://www.ngo-online.de/lexikon/kilowatt-peak-kwp-spitzenleistung

[30] "Stromverbrauch von Waschmaschinen," (Zugriff: 27.10.2016). [Online]. Available: http://www.stromverbrauchinfo.de/stromverbrauch-waschmaschinen.php

[31] "Energieverbrauch von Wäschetrocknern," *Verbraucherzentrale Rheinland-Pfalz e.V. and Öko-Institut e. V., Institut für angewandte Ökologie*, 09/2012, (Zugriff: 27.10.2016). [Online]. Available: http://www.die-stromsparinitiative.de/fileadmin/dokumente/PDF/infoblatt_trockner_druck.pdf_01.pdf

[32] "Warmwasser-Wärmepumpe," (Zugriff: 27.10.2016). [Online]. Available: http://www.energie-experten.org/heizung/waermepumpe/warmwasser-waermepumpe/

[33] "Jahresarbeitszahl," (Zugriff: 27.10.2016). [Online]. Available: http://www.energie-experten.org/heizung/waermepumpe/leistung/jahresarbeitszahl.html

[34] "Stromverbrauch von Warmwasser-Wärmepumpen," 19.08.2015, (Zugriff: 27.10.2016). [Online]. Available: http://www.energie-experten.org/heizung/waermepumpe/warmwasser-waermepumpe/stromverbrauch.html

[35] E. Heindl, "Lithium oder Blei Batteriespeicher, ein Vergleich," 28.10.2016, (Zugriff: 07.11.2016). [Online]. Available: http://energiespeicher.blogspot.de/2013_06_01_archive.html

[36] J. Weniger and V. Quaschning, "Begrenzung der Einspeiseleistung von netzgekoppelten Photovoltaiksystemen mit Batteriespeichern," in *28. Symposium Photovoltaische Solarenergie*, März, 2013.

[37] G. van Rossum and F. L. Drake Jr, *Python Tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.