Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

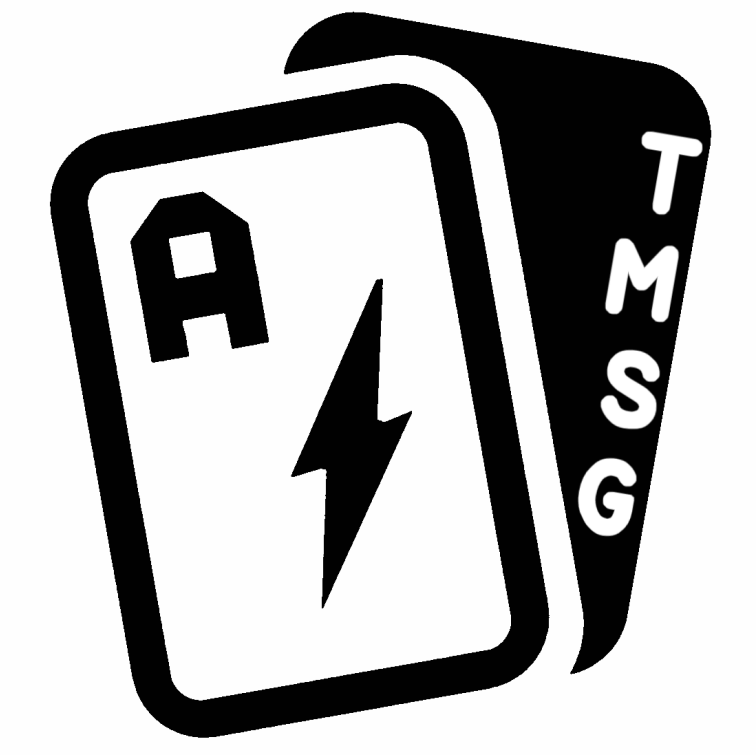
Department of the Environment,
Transport, Energy and Communication DETEC

Swiss Federal Office of Energy SFOE
Energy Research and Cleantech

Final report

Training Multiagent Strategies for the Grid

Final Report



University of Applied Sciences and Arts
of Southern Switzerland

SUPSI

Date: 03/11/2022

Place: Bern

Publisher:
Swiss Federal Office of Energy SFOE
Research Programme SWEET
CH-3003 Bern
www.bfe.admin.ch
energieforschung@bfe.admin.ch

Author:
Federico Rosato, ISAAC, SUPSI (federico.rosato@supsi.ch)

SFOE head of domain: Dr. Laura Ding, laura.ding@bfe.admin.ch
SFOE programme manager: Dr. Laura Ding, laura.ding@bfe.admin.ch
SFOE contract number: SI/502274

Summary

The *Training Multiagent Strategies for the Grid* project aims to tackle the problem of modeling electrical flexibilities in the distribution grid, and then to provide sensible strategies for autonomous agents that manage those flexibilities. We model this environment as a day-long strategic interaction among the agents, which try to optimize the benefit to the final user they represent while not having a disruptive impact on the grid through excessive coordination. The agents take into account a daily electricity price curve in their optimization, unlocking the potential to use the price signal as a flexibility steering tool. This background lends itself to the usage of the Stochastic Game framework, a form of dynamic game in which the joint actions of an array of agents determine the evolution of the environment. The Stochastic Game is a standard framework for the study of Multiagent Reinforcement Learning, and we adopt techniques from this field to try and propose sensible solutions. Given the particular structure of our problem, we propose a new two-level method for the calculation of strategies, which proves effective. We analyze the method and perform a simulation campaign on the output strategies, analyzing the effect of various parameters that characterize the model, and evaluating the potential in steering flexibility through changes in the daily price curve. Finally, in order to evaluate the effect of the agents on the grid, a power flow simulation is run alongside the agents, and the effect of various strategies on the voltage distribution is analyzed.

Zusammenfassung

Die *Training Multiagent Strategies for the Grid* Projekt zielt darauf ab, das Problem der Modellierung elektrischer Flexibilitäten im Verteilungsnetz anzugehen und anschließend sinnvolle Strategien für autonome Agenten bereitzustellen, die diese Flexibilitäten verwalten. Wir modellieren diese Umgebung als eine ganztägige strategische Interaktion zwischen den Agenten, die versuchen, den Nutzen für den Endverbraucher, den sie repräsentieren, zu optimieren und gleichzeitig keine störenden Auswirkungen auf das Netz durch übermäßige Koordination zu haben. Die Agenten berücksichtigen bei ihrer Optimierung eine tägliche Strompreiskurve und erschließen so das Potenzial, das Preissignal als Flexibilitätssteuerungsinstrument zu nutzen. Vor diesem Hintergrund bietet sich die Verwendung des Stochastic Game Frameworks an, einer Form des dynamischen Spiels, bei dem die gemeinsamen Aktionen einer Reihe von Agenten die Entwicklung der Umwelt bestimmen. Das stochastische Spiel ist ein Standardrahmen für die Untersuchung des Multiagenten-Verstärkungslernens, und wir übernehmen Techniken aus diesem Bereich, um zu versuchen, sinnvolle Lösungen vorzuschlagen. Angesichts der besonderen Struktur unseres Problems schlagen wir eine neue zweistufige Methode für die Berechnung von Strategien vor, die sich als effektiv erweisen. Wir analysieren die Methode und führen eine Simulationskampagne für die Output-Strategien durch, wobei wir die Auswirkungen verschiedener Parameter, die das Modell charakterisieren, analysieren und das Potenzial für die Steuerung der Flexibilität durch Veränderungen der täglichen Preiskurve bewerten. Um schließlich die Auswirkungen der Agenten auf das Netz zu bewerten, wird eine Leistungsflusssimulation parallel zu den Agenten durchgeführt und die Auswirkungen der verschiedenen Strategien auf die Spannungsverteilung analysiert.

Résumé

Le projet *Training Multiagent Strategies for the Grid* vise à résoudre le problème de la modélisation des flexibilités électriques dans le réseau de distribution, puis à fournir des stratégies raisonnables pour les agents autonomes qui gèrent ces flexibilités. Nous modélisons cet environnement comme une interaction stratégique d'une journée entre les agents, qui tentent d'optimiser le bénéfice pour l'utilisateur final qu'ils représentent tout en n'ayant pas d'impact perturbateur sur le réseau par une coordination excessive. Les agents tiennent compte d'une courbe quotidienne des prix de l'électricité dans leur optimisation, ce qui permet d'utiliser le signal de prix comme outil de pilotage de la flexibilité. Ce contexte se prête à l'utilisation du cadre du Jeu Stochastique, une forme de jeu dynamique dans lequel les actions conjointes d'un ensemble d'agents déterminent l'évolution de l'environnement. Le Jeu Stochastique est un cadre standard pour l'étude de l'Apprentissage par Renforcement Multi-agents, et nous adoptons des techniques de ce domaine pour essayer de proposer des solutions raisonnables. Compte tenu de la structure particulière de notre problème, nous proposons une nouvelle méthode à deux niveaux pour le calcul des stratégies, qui s'avère efficace. Nous analysons la méthode et effectuons une campagne de simulation sur les stratégies de sortie, en analysant l'effet des différents paramètres qui caractérisent le modèle, et en évaluant le potentiel de flexibilité de pilotage par le biais de changements dans la courbe des prix quotidiens. Enfin, afin d'évaluer l'effet des agents sur le réseau, une simulation de flux de puissance est exécutée avec les agents, et l'effet de différentes stratégies sur la distribution de la tension est analysé.

Sommario

Il progetto *Training Multiagent Strategies for the Grid* si propone di affrontare il problema della modellazione delle flessibilità elettriche nella rete di distribuzione e di fornire strategie adeguate per agenti autonomi a cui viene affidato il compito di gestire tali flessibilità. Questo ambiente viene modellato come un'interazione strategica giornaliera tra gli agenti, che cercano di ottimizzare i benefici per l'utente finale, senza avere un impatto distruttivo sulla rete a causa di un'eccessiva coordinazione. Gli agenti tengono conto di una curva di prezzo giornaliera nella loro ottimizzazione, offrendo la possibilità di utilizzare il segnale di prezzo come strumento di gestione della flessibilità. Questo contesto si presta ad essere modellato come Stochastic Game, una forma di gioco dinamico in cui le azioni congiunte di una serie di agenti determinano l'evoluzione dello stato globale. Lo Stochastic Game è un framework standard per lo studio del Multiagent Reinforcement Learning e adotteremo le tecniche di questo campo per cercare di proporre soluzioni adeguate. Data la particolare struttura del nostro problema, viene proposto un nuovo metodo a due livelli per il calcolo delle strategie, che si rivela efficace. Il metodo viene analizzato e viene eseguita una campagna sperimentale sulle strategie calcolate, analizzando l'effetto di vari parametri che caratterizzano il modello e valutando il potenziale di controllo della flessibilità attraverso le variazioni della curva dei prezzi giornaliera. Infine, per valutare l'effetto degli agenti sulla rete, viene eseguita una simulazione elettrica includente gli agenti e viene analizzato l'effetto delle varie strategie sulla distribuzione di tensione.

Contents

List of abbreviations

BSP	Balancing Services Provider
EV	Electric Vehicle
HMM	Hidden Markov Model
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Process
PV	Photovoltaic
RL	Reinforcement Learning
V2G	Vehicle-to-Grid

Part I

Introduction

In the TMSG project, we aim to study the behavior of smart, automatic agents managing electrical flexibilities in the context of an electrical distribution grid, seeking optimal customer satisfaction while taking into account the effect of their collective actions on the grid. The conceptual tools used for accomplishing these results will be the ones typical of Multiagent Reinforcement Learning (MARL), even if we seek solution models not based on function approximation. In order to make them applicable, the first task to perform is to formulate the real situation as a mathematical model suitable for the application of these techniques. One of the most common settings used in this area, and one that will be favorable for our situation, is the Stochastic Game. Therefore, we will begin by providing a brief review of Stochastic Games and a general, high-level description of the situation we want to capture, before laying out exactly the formalization.

The central object of study of this project is the group of electrical appliances and electrical demand profiles pertaining an individual residential owner, using the distribution network. Typically, this will coincide with a single household, building, et cetera, comprising electrical appliances such as PV panels, stationary batteries, EVs, heat pumps for building heating and stochastic electrical demands corresponding to the final user's discretionary electrical consumption. Appliances and profiles pertaining a single owner should be managed together in order to achieve the owner's best interest by leveraging, potentially, synergies and interactions between them; and in fact, as we will see in the following sections, in the end we will have a single automated agent "representing" an owner, managing her appliances. The electrical appliances considered have very different characteristics. PV panels, for example, have essentially aleatory production directly linked to the weather, that in turn can be forecasted stochastically with time-varying accuracy. EVs have batteries that, given that Vehicle-to-Grid (V2G) technology is present, can be steered in both the consumption and absorption direction within the SOC limits, but with the important limitation that the user wants to find a charged vehicle when she needs it, and this mobility need appears in a stochastic way, partially foreseeable according to the user's habits. Heat pumps for building heating can be turned on and off, but suffer from excessive start-and-stop. In very general terms they have to keep charged a working fluid buffer that decouples them from the thermal loads and the heating system can use the environment as thermal "storage" within certain comfort temperature limits, so they present a certain degree of flexibility. Nevertheless, user discomfort appears if the temperature limits are violated. Stationary batteries, frequently installed together with PV panels as buffers, are completely steerable without influencing the user's needs. The presence of various flexibilities with different operating characteristics paints a complex scenario where techniques capable of tackling their optimization in an orchestrated way are required.

One key aspect that makes steering flexibilities in a distribution grid interesting and challenging is the presence of other agents that will have similar self-interests and similar means to seek their optimization. Since external conditions, such as the weather, are essentially the same for the agents in a local distribution grid, agents representing final users with a similar set of appliances and similar needs and habits will tend, if unconstrained, to coordinate significantly their control actions (net a certain amount of statistical variability). Coordinated actions (absorption/injection) would translate into peaks of stress on the grid. Since one of the main targets of innovative demand response and flexibility management techniques is, in general, to enable the distribution grid to accept new loads and generation with minimal material overhauls, this aspect is of capital importance and must be contemplated during the design and study of the agents' behavior, incentivizing them not to correlate excessively their behavior.

1 Context and state of the art of research

The situation described above poses complex strategical questions that need to be inscribed in a well-defined framework to be tackled. In this work we are interested in modeling, designing and analyzing the strategic behavior of agents that seek to maximize their own benefit inside the described environment; to do so, we will construct a formalization of the situation above as a Stochastic Game. Stochastic Games are a powerful and comprehensive framework, capable of modeling stateful, long term multi-period strategic situations involving multiple self-interested agents. Importantly, they are a generalization of both Markov Decision Processes (MDPs), a central object in Dynamic Programming and Reinforcement Learning, and Repeated Games, coming from Game Theory. They possess, in fact, both the aspects of long-term cumulative reward maximization present in MDPs and the ones of multi-stage strategic interaction with other players of Repeated Games. Very informally, they can be seen as a "union" of the two models; not surprisingly, they are a central modeling tool in Multi Agent Reinforcement Learning (MARL) [1], the field that seeks to extend RL to a multiagent setting and whose techniques we will use in this work. The classic paper on Stochastic Games is due to Shapley [2]. Since they live at the intersection of DP/RL and Game Theory, milieu, terminology and concepts of both worlds are important to gain the ability to work fully and meaningfully with them. Therefore, in the following subsections we will introduce both of these sides, and see how SGs can be obtained as a generalization.

1.1 Markov Decision Processes

Reinforcement Learning is concerned with methodologies to find sensible behavior for agents operating in a potentially stochastic environment, able to make choices that influence the state of the environment and subject to receiving rewards according to its evolution. RL is a wide topic, with literature spanning decades of work [3]. In its latest incarnations, it has received significant attention in the last years, due to a string of successes in producing behavior policies capable of guiding the actions of agents in very complex scenarios, often achieving superhuman performance [4, 5]. The roots of RL can be traced back to Dynamic Programming, a set of classical, powerful techniques introduced in the 1950s by Bellman for solving multi-period optimization problems, and conceptually descend from the eponymous principle:

> *Bellman's Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. [6]*

The basic concrete stochastic framework in which the Bellman Optimality Principle can be translated quantitatively into a condition for optimal behavior is the Markov Decision Process (MDP).

A (discounted) MDP can be defined as a tuple (S, A, P, R, γ) where

- $S = \{s_1, ..., s_q\}$ is a set of q states, or *state space*;
- $A = \{a_1, ..., a_m\}$ is the set of m actions available to the agent, or *action space* (in general, it might also be that $A = A(s)$, that is that the set of actions available might be different from state to state, but we will not use this more general setting);
- $P = P(s'|s, a)$ is a function $P : S \times S \times A \to [0, 1]$ indicating the probability of going from the present state s to state s' when action a is performed;

- $R = R(s, a)$ is a function $R : S \times A \to \mathbb{R}$ indicating the immediate reward experienced by the agent when in the present state s it performs action a;
- $\gamma \in [0, 1]$ is a *discount factor*, indicating that rewards experienced in the future are subject to a discount; that is, the agent at time τ values a reward experienced in the future at time t as $\gamma^{(t-\tau)}R$.

Agents in an MDP can act by following a *policy*, that is a rule $\pi(s)$, $\pi : S \to A$ prescribing what behavior to take at each state the agents finds itself in. With reference to an agent following a fixed policy $\bar{\pi}$, a function $V^{\bar{\pi}} = V^{\bar{\pi}}(s)$ can be defined representing the expected discounted cumulative reward experienced by the agent, or, more succintly, the *value* of state s for the agent. Since at each state $V(s)$ represents the expected cumulative reward *from that state onwards*, it is possible to write the following recursive relationship:

$$V^{\bar{\pi}}(s) = \mathbb{E}\big(\sum_{t=0}^{\infty} \gamma^t R_{t+1}\big) = \mathbb{E}\big(R(s, \bar{\pi}(s)) + \gamma V^{\bar{\pi}}(s')\big) \tag{1}$$

$$V^{\bar{\pi}}(s) = R(s, \bar{\pi}(s)) + \sum_{s'} P(s'|s, \bar{\pi}(s))\gamma V^{\bar{\pi}}(s') \tag{2}$$

It is useful to introduce function Q, which stands for *quality function*, representing the value of taking action a in state s when playing policy $\bar{\pi}$ forever after:

$$Q^{\bar{\pi}}(s, a) = R(s, a) + \sum_{s'} P(s'|s, a)\gamma V^{\bar{\pi}}(s') \tag{3}$$

so that:

$$V^{\bar{\pi}}(s) = Q^{\bar{\pi}}(s, \bar{\pi}(s)) \tag{4}$$

This notation is useful not only because it allows to write equations more compactly, but because the q function encapsulates all information about transition probabilities and rewards, and this is useful in situations in which P and R (often referred to as the *model* of the environment) are not known, in which case an attempt can be made to learn the q function directly with a wide array of method; the most basic, tabular Q-learning, proceeds by interacting with the environment, experimenting directly the effect of taking action a in state s without possessing an accurate knowledge of the "mechanics" of the MDP being played beforehand. In this work, though, we are mostly concerned with solving Stochastic Games with a known model.

Now suppose that there exists a policy, dubbed *optimal* and indicated with π^*, with the property that

$$V^{\pi^*}(s) >= V^{\pi}(s) \; for \; any \; \pi, s \tag{5}$$

Bellman's Principle of Optimality, in this framework, translates into a necessary criterion for the optimal policy π^*:

$$V^{\pi^*}(s) = \max_{a \in A(s)} Q^{\pi^*}(s, a) \tag{6}$$

It can be shown that an optimal policy exists. A way of doing so can be sketched in the following way. First of all we build a "Bellman Optimality Operator" $\mathcal{B} : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ based on Equation (6) such that:

$$\mathcal{B}V(s) = \max_{a \in A(s)} R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s') \tag{7}$$

This operator encapsulates the information contained in fixed parameters P, R, γ. It takes a vector of dimension $|S|$, which intuitively is the real vector of state values, and maps it to an updated vector of similar nature, residing in the same space. It can then be shown in a fairly straightforward manner that this vector operator is a contraction map with coefficient γ which, by Banach's fixed point theorem, has a unique fixed point. It is easy to show (by the very definition of the operator, in fact) that this fixed point will satisfy Equation (6), and is therefore the optimal policy. A proof by the fixed point theorem is constructive, and in fact yields an immediate way of finding the state values under optimal policy, which is the repeated application of 7 on a vector of state estimates, however initialized. This method is the heart of the algorithm commonly known as *Value Iteration* (Algorithm 1). The policy can be then recovered by applying a greedy strategy over the found vector of values, i.e., by Equation (6).

Algorithm 1 MDP Value Iteration

procedure VALUEITERATION(S, A, P, R, γ)
  Initialize $V(s) = 0, V'(s) = 0$
  while V *is not converged* **do**
    for $s \in S$ **do**
      $V'(s) \leftarrow \max_a Q(s,a) = \max_a \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s') \right)$
      $V(s) \leftarrow V'(s)$
  return V

Given again a fixed policy $\bar{\pi}$ and recalling Equation 2, it is possible to define a *Bellman Evaluation Operator* $\mathcal{B}$, that works similarly to the Bellman Optimality Operator. However, since it integrates the selection of the action dictated by the fixed policy instead of the one maximizing expected value, its fixed point is the vector of state values induced by the fixed policy, not the optimal one.

$$\mathcal{B}V(s) = R(s, \bar{\pi}(s)) + \sum_{s'} P(s'|s, \bar{\pi}(s)) \gamma V^{\bar{\pi}}(s') \tag{8}$$

Another common method is *Policy Iteration* (Algorithm 2). This method is slightly more complex, but it directly retrieves the optimal policy, it converges faster, in fewer iterations and yields the exact solution (whereas value iteration converges exponentially to the solution but formally never reaches it).

Algorithm 2 MDP Policy Iteration

procedure POLICYITERATION(S, A, P, R, γ)
  Initialize $\pi(s), \pi'(s)$ randomly. **while** π *is not converged* **do**
    Find V^π using Equation (2) **for** $s \in S$ **do**
      $\pi'(s) \leftarrow \operatorname{argmax}_a Q^\pi(s,a) = \operatorname{argmax}_a \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a)V^\pi(s') \right)$
    $\pi(s) \leftarrow \pi'(s)$
  return π

1.2 Stochastic Games as a generalization of MDPs

Stochastic Games with n players can be obtained as a straightforward generalization of MDPs. The intuitive idea is that instead of feeding a single action to the environment, the n players chose at the same time what action to perform, and the whole list of action is fed to the environment, that evolves to a new state and rewards individually the players. From a formal standpoint, it is sufficient to substitute the action space A with a set of *vectors* of actions having length n. Therefore, instead of a single action a we will have an action vector $\mathbf{a} = [a_1, ..., a_n]$. The reward function R becomes a function with vectorial values. The rest of the definition is essentially unmodified.

- $S = \{s_1, ..., s_q\}$ is a set of q states, or *state space*;
- $\mathbf{A} = \{\mathbf{a}_1, ..., \mathbf{a}_m\}$ is the set of m joint actions that the agents can jointly perform, or *action space*. If the cardinalities of the sets of actions available to each of the n agents are $m_1, ..., m_n$, the total cardinality of the vectorial action space is then $\prod m_i$. It might also be the case that the actions available to the agents individually depend on the state, but we will not use this more general setting.
- $P = P(s'|s, \mathbf{a})$ is a function $P : S \times S \times \mathbf{A} \to [0, 1]$ indicating the probability of going from the present state s to state s' when action a is performed;
- $\mathbf{R} = \mathbf{R}(s, \mathbf{a})$ is a function $\mathbf{R} : S \times \mathbf{A} \to \mathbb{R}^n$ indicating the vector of immediate rewards individually experienced by the agents when in the present state s they perform the joint action $\mathbf{a}$;
- $\gamma \in [0, 1]$ is a *discount factor*, indicating that rewards experienced in the future are subject to a discount; that is, the agent at time τ values a reward experienced in the future at time t as $\gamma^{(t-\tau)}R$.

In general, policies for Stochastic Games use the whole A set as support, that is, they are mixed strategies specifying probabilities over the available actions at each state s for each player $h \in 1, ..., n$. We will use the notation $x = [x^h_{s,a}]$ to indicate a joint policy for a Stochastic Game. Quantities like the value of a state given a policy x will now be vectors of length n; we will indicate the value of state s for player h given the joint policy x as $V^x(s, h)$.

1.3 Stochastic Games as a generalization of Repeated Games

At first sight the introduction of the generalization above seems very straightforward, but the potential interaction among the agents encoded in P and $\mathbf{R}$ introduces the full strategic complexity that is typically studied in game theory [7]. We can notice, in fact, that Stochastic Games are a generalization of Repeated Games. In their common acception, n-player Repeated Games consist in the repeated execution of a single basic *stage game*, in which the n players each choose an action from their action space and receive an individual payout according to joint action they formed. The natural objective of the player is then to maximise some cumulative payout formula, often discounted. This situation can be interpreted as a 1-state Stochastic Game, in which the probability function P encodes trivially the "transition" from the game to itself with probability 1, and the reward function encodes the payouts of the single stage game. If we introduce the following generalization:

- The stage game being played can change, being chosen from a set of n stage games;
- The transition from stage game s to stage game s' is decided stochastically, according to a function that depends on the joint action of the players;

- The payout to the players is decided according to a function that depends on the joint action of the players;

then we again get a Stochastic Game. An important observation to make is that, in repeated games, for the strategies to be nontrivial (and in fact equivalent to strategies for the stage game only) they should depend on the history of play, that is, the player should have memory of past play history and be able to act upon it. In a Stochastic Game, it is interesting to consider strategies that depend only on the state (the game being played), which are labelled *stationary strategies* [8], in a similar fashion to policies for an MDP. It would also be possible to consider *behavioral strategies* depending on the history of play, but this task, in a realm like Stochastic Games in which stationary strategies are already not easy to find, is often intractable.

1.4 Algorithms for Stochastic Games

The literature on Stochastic Games is vast [9] and encompasses 6 decades of research. From the generality of the model descends a faceted taxonomy, with research alleys and many results concentrating on specific cases such as 2-player or zero-sum (that is, the sum of the total rewards for the player is always zero). Expectedly, the additional structure of the special cases helps in finding additional and stronger results and efficient algorithms for finding optimal strategies of the players, but in this work we are inherently interested in the general case with n players and general sum, since a grid with only 2 agents managing flexibilities would be rather uninteresting and the reward structure is not zero-sum, in that players might perceive incentive payouts, customer satisfaction and dissatisfaction, or penalties totaling any value, according to the particular market and rule structure chosen. Some of the proposed algorithms with convergence guarantees, such as [10,11] are learning procedures that, despite easy applicability, only partly leverage the known structure of the game and therefore necessitate long times of information accumulation. Theoretically sound procedures such as [12] rapidly become intractable even at medium sizes. These algorithms have great theoretical importance, but rapidly become unwieldy for real case studies commanding a minimally sizeable state space. One interesting algorithm is given by Akchurina [13] and consists, at its heart, in the extension of Replicator Dynamics [14] to the context of Stochastic Games. Albeit encountering fluctuations and slow convergence in the first test with random games, even intermediate solutions by this algorithm could provide useful insights, which is the main aim of this work. This approach, though, has strong limitations in terms of the state space size that can be addressed, realistically limited to a few hundreds. In the subsequent phases of the work, though, a more promising approach appeared to be the one exposed in Section 5, leveraging more deeply the particular advantages given by the structure of the problem studied in Section 4 and allowing more promisingly states spaces that near 10^5 states for a single agent in a regime of imperfect information, therefore with a global state space of very significant size.

1.5 Regularized and reward-robust MDPs

A recent work by Geist et al. [15] formulates a general theory of Regularized MDPs. These models make use of *regularization*; this means that, with respect to the standard (dubbed in this context "unregularized") MDP model, an additional policy-dependent term (the *regularizer*) is added to the evaluation operators in order to "skew" the resulting policies in such a way that the optimization of this term is taken into account in addition to the bare rewards.

Indicating the state space with S as done previously, the regularizer has the form $\Omega(\pi)$, which can be

seen as a vector of length $|S|$ containing the values of just as many functions having the policy at state s as argument, $\Omega_s(\pi_s)$. Recalling equation 8, the Ω-regularized Bellman Evaluation Operator $\mathcal{B}_{\bar{\pi},\Omega}$ is defined in the following way:

$$\mathcal{B}_{\bar{\pi},\Omega}V(s) = \mathcal{B}_{\bar{\pi}}V(s) - \Omega(\bar{\pi}) \tag{9}$$

We can then define a Ω-regularized Bellman Optimality Operator $\mathcal{B}^*_\Omega$ in the following way:

$$\mathcal{B}^*_\Omega V(s) = \max_\pi \mathcal{B}_{\pi,\Omega}V(s) \tag{10}$$

The $\mathcal{B}^*_\Omega$ operator mantains the same key favorable property of its unregularized counterpart, that is, it is a γ-contraction in supremum norm whose fixed point is the optimal, regularized policy, but under one condition: the regularizer Ω must be strongly convex. This means that value-based methods provably converge in general using this scheme only under this convexity requirement.

Incidentally, we notice that, differently from the unregolarized optimality operator, the optimization variable is the whole vectorial policy, π. This generalization is necessary in that the optimal regularized policy might be mixed. In the unregularized case, the optimal policy is always pure, and therefore in Equation 7 the optimization variable is the individual action a. It would be possible to use this more general formulation in the unregularized case, too, but the vector representing the optimal policy at each state would always be of the form $0...0, 1, 0, ...0$.

In a successive work, Derman, Geist and Mannor investigate the relationship between regularized MDPs and robust MDPs [16], that is, MDPs in which a solution is sought under uncertain parametrizations, optimizing against a worst-case scenario. In particular, they arrive to identify an equivalence between regularized MDPs with *reward-robust* MDPs (that is MDPs in which there is uncertainty only on the reward). To use this framework, the reward uncertainty has to be represented as a support function, and therefore be strongly convex, which is expected for the equivalence with the above framework to be valid.

At a first glance, these conceptual tools seem promising. As observed previously in this chapter, by considering symmetrical agents, the Stochastic Game being played can be seen as an MDP played individually by the agents in which the reward experienced has an additional "external" component (see Equation 25) due to the energy consumption coordination, which is dependent on the policy through the expected energy consumption. The coordination component could, therefore, be formally interpreted as a policy regularizer. Equivalently, the policy must be "robust" against the coordination effect caused by the policy itself. Unfortunately, under our formulation, there are several difficulties:

- The regularizer functions considered in the cited work are fixed, whereas we would need a regularizer function whose shape is itself dependent on the policy. This aspect makes the framework way more complicated to analyze theoretically, barring the possibility to use conventional proofs based on contraction properties when building a regularized Bellman Optimality Operator.
- When considering just a Bellman Evaluation Operator, the policy is fixed and therefore the regularizer is fixed too. It is possible to prove that the Bellman Evaluation Operator has the contraction property, but the fixed regularizer would in general be non convex, a property needed to guarantee the uniqueness of the solution. In fact, even in the simple case of actions having a fixed collective impact represented by a vector ξ, the average impact of the player would have the form:

$$f(\pi) = \xi \cdot \pi \cdot p^\infty(\pi) \tag{11}$$

Where p^∞ is the stationary state distribution obtained by playing fixed strategy π. We can study this expression in a more detailed way. First of all, instead of considering f a function of the policy, let's fix a starting policy π_0 and a second policy π_f, and consider "interpolated" policies dependent on a parameter $t \in [0,1]$ of the form $x(t) = (1-t)\pi_0 + t\pi_f$, that is a convex combination of the two. When fixing a policy, the MDP reduces to a Markov Chain with a transition matrix $P_\pi(s'|s)$ whose elements are the convex combination $\sum_a P(s'|s,a)\pi_{s,a}$. If considering the above mentioned form of policy paramterized on t, the transition matrix becomes itself dependent on the parameter: $P(t) = (1-t)P_0 + tP_f$. We also introduce the perturbation matrix $F(t) = P(t) - P_0$. According to [17], there is an explicit expression for the stationary distribution of a perturbed ergodic Markov Chain, that is an ergodic Markov Chain whose transition matrix is changed by the addition of a perturbation matrix:

$$p^\infty(\pi(t)) = p^\infty(t) = p_0^\infty \cdot F^\star(t) \tag{12}$$

$$F^\star(t) = (I - F(t) \cdot (I - P_0)^\#) \tag{13}$$

where $A^\#$ is the Drazin inverse of A [18], whose central role in stochastic models is well-known [19–21]. With the aid of this expression, we can compute the directional derivatives of $f(x(t)) = f(t)$, that is the derivative of f calculated at policy x_0 along the vector $x_f - x_0$.

$$f'(t) = \xi \frac{\partial \pi(t)}{\partial t} p_0^\infty F^\star(t) - \xi \pi p_0^\infty F^\star(t) \frac{\partial F(t)}{\partial t} F^\star(t) \tag{14}$$

The $\frac{\partial \pi(t)}{\partial t}$ derivative is constant and is equal to $\pi_f - \pi_0$. The $\frac{\partial F(t)}{\partial t}$ is equal to $\frac{\partial P(t)}{\partial t}$ and is similarly constant and equal to $P_f - P_0$.

$$f''(t) = -2\xi(\pi_f - \pi_0)p_0^\infty F^\star(t)(P_f - P_0)(I - P_0)^\# F^\star(t) + 2\xi\pi(t)p_0^\infty F^\star(t)\left[(P_f - P_0)(I - P_0)^\# F^\star(t)\right]^2 \tag{15}$$

When creating a smooth regularizer $\Omega(f(\pi))$, sufficient conditions for its convexity based on the positiveness of the second derivative would in general depend on this expression. Even if Ω itself is convex, the "easy" sufficient condition for $\Omega \circ f$ to be convex without further inspection would be for f to also be convex, and therefore f'' to be positive whatever the choice of π_0 and π_f. It is challenging to find conditions to pose on the $P(s'|s,a)$ matrix that would ultimately result in expression 15 to be certainly positive. Nevertheless this remains an interesting open question.

- This framework does not take into account the information structure, and in particular the fact that the state of the solving agent conditions the distribution of the state of the other agents, an aspect that will be discussed in detail later.
- As shown in Chapter 3, the actual collective penalty term does not depend directly on the action taken, but on the absorbed energy $\mathcal{E}$, which depends on the next state. This means that we need to penalize this term, and not the actions themselves, to have a coherent solution.

This is, interestingly, an effect of the strong nonstationarity of learning processes generally found in Stochastic Games: changing the policy has a complex effect on the reward experienced.

1.6 Constrained MDPs

It is known that MDPs can be formulated equivalently as Linear Programs [22]. One possible formulation is the following:

$$\begin{aligned} \max_{\lambda} \quad & \sum_{s}\sum_{a}\lambda_{s,a}R(s,a) \\ \text{s.t.} \quad & \sum_{a}\lambda_{s,a} = \sum_{s}\sum_{a}\lambda_{s,a}P(s'|s,a) \\ & \sum\lambda = 1 \\ & \lambda >= 0 \end{aligned} \tag{16}$$

where $\lambda_{s,a}$ is the long-term portion of time spent in state s choosing action a; in other words, λ is a matrix similar the policy, whose value is the policy itself with each row multiplied by the value of the stationary distribution of the corresponding state.

$$\lambda_{s,a} = x_{s,a}p^{\infty}(s) \tag{17}$$

The objective is, as always, the maximization of reward. The first constraint is a kind of "conservation law" stating that the permanence in state s must be equal to the sum of all the (s,a) pairs multiplied by the probability of the pair leading to s, which is exactly the information contained in matrix P. The second and third constraint simply ensure that

A central advantage of the LP formulation is the ability to conveniently extend the model with additional linear constraints [23, 24]. If, as in the previous section, we call $\xi(a)$ the collective impact of action a, we can directly pose a limit $\bar{\xi}$ on the average impact caused by the individual agent with the following extended formulation:

$$\begin{aligned} \max_{\lambda} \quad & \sum_{s}\sum_{a}\lambda_{s,a}R(s,a) \\ \text{s.t.} \quad & \sum_{a}\lambda_{s,a} = \sum_{s}\sum_{a}\lambda_{s,a}P(s'|s,a) \\ & \sum_{s}\sum_{a}\lambda_{s,a} = 1 \\ & \lambda_{s,a} >= 0 \\ & \sum_{s}\sum_{a}\lambda_{s,a}\xi(a) <= \bar{\xi} \end{aligned} \tag{18}$$

The resulting model is a formulation of Constrained Markov Decision Problems (CMDPs). The standard theoretical treatise for CMDPs is [25], in which numerous variations are explored. Each action impact could also be state-dependent while preserving the nature of the optimization model, but only in a predetermined way. Therefore, albeit this formulation is powerful, it is not directly applicable to our case, or, more precisely, it would be applicable but by accepting an intrinsic inefficiency source that is difficult to quantify, due to discarding the additional information. There is no obvious way to integrate the additional information structure in the optimization, which depends on the whole policy. As an informal example, an agent in state s could infer due to its s_{sh} a probability distribution for the state of the other agents which, according to the current strategy employed by the community, results in a low expected

energy absorption $\hat{\mathcal{E}}$ (these details will be explained more deeply in Part II). In this situation, the agent could afford actions that cause higher energy absorption for itself. If the public information is discarded and p^{∞} is used instead of the more accurate distribution, it may choose a different, less optimal action. As we will see, the original method we will propose in the following part of the chapter accounts for this effect. Finding the conditional distribution is a problem of considerable size, in fact it corresponds to solving a Markov Chain describing the joint evolution of two agents, the size of the action space of which is therefore squared w.r.t. the base problem. Even if a formulation as optimization problem was attainable, the scale of the problem would increase substantially, realistically placing it out of the reach of even modern optimization solvers. For a P like the one in Figure 6 and the subset subdivision seen in Section 4.2, the joint transition matrix would have a number of nonzero elements of order just shy of 10^6, as we will see in detail in Section 6. Nevertheless, such formulations remain a very interesting topic for future work, owing to the theoretical guarantees they would bring.

1.7 POSGs and FOSGs

Another theoretical specialized framework, particularly relevant for our situation is the one of *Partially Observable Stochastic Games*, or POSGs [26]. POSGs are a generalization of Stochastic Games that take into account the fact that the players might not have complete information about the "world" state. In addition to the definition tuple of Stochastic Game $\mathcal{G}$ provided in Section 1.2, a POSG has the following elements:

- $\{\mathcal{O}_i\}$, a collection of i sets (one per agent), $\mathcal{O}_i = \{o_1, ..., o_j\}$ representing the possible distinct observations that can be made by the agent. A joint observation is indicated by $\mathbf{o} \in \times_i \mathcal{O}_i$
- $O = O(\mathbf{o}|s, \mathbf{a})$, an observation function $O : S \times \mathbf{A} \to [0, 1]$ indicating the probability of realizing the joint observation $\mathbf{o}$ when performing joint action $\mathbf{a}$ in state s.

This framework models well the partiality of information available the agents. Unfortunately, the already high theoretical computational difficulty of SGs are further exacerbated in POSGs, which are NEXP-hard [27], due to the fact that beliefs have to be mantained by the players about the true state. A few research works have been dedicated to special cases of POSGs [28–30]. Of particular interest is a recent work of Horàk [31], which makes use of Public Observations, which further resembles our problem comprising a public part of the state as explained in Section 4.1, but is limited to the 2-player, constant-sum case.

A very interesting further generalization of POSGs are *Factored-Observation Stochastic Games* or FOSGs, introduced in a recent work by Kovařík et al. [32]. In this framework, the imperfect information structure mediated by observation functions includes explicitly a public and a private part. With reference to the notation introduced in 1.2, the two key generalizations that modify a POSG into a FOSG are the following:

- There is a "player function" $p : S \to \mathbf{2}^q$ that associates to each state a subset of agents that can perform an action;
- The space of joint observations is factored with one additional set $\mathcal{O}_{PUB}$, that represent a public observation available to all agents. therefore, $\mathbf{o} \in \mathcal{O}_{PUB} \times (\times_i \mathcal{O}_i)$.

This formalization has the advantage of explicitly considering information that is universally available, which is relevant as the fact that agents (as it happens in many real examples) can limit their reasoning

to situations that are compatible with the public knowledge. For example, in our setting the state of the weather and the time are a publicly available part of the state, and the agents should condition their reasoning, their belief on the state of the other agents and, ultimately, their strategies on this information. The "player function" allows for generalizations in the time and "turn" structure of the game which open interesting convergences with methods for the solution of Extensive Form Games, but here we are less interested on this aspect. General-case FOSGs, naturally, inherit the complexity of POSGs, as POSGs are a special case of FOSGs in which p trivially indicates all agents at all states and $\mathcal{O}_{PUB} = \emptyset$. Nevertheless, our problem can be described as a FOSG with the critically simplifying assumption of symmetrical agents, but even in this case, to the best of the author's knowledge the literature on solution methods for this framework is scarce (the computational difficulties for principled solutions were underlined and furthermore the definition of the framework is very recent), let alone for large state spaces.

2 Unconventionality and objectives

The approach proposed here involves the evaluation of cutting-edge techniques for training agents in multiagent settings. The introduction of active and steerable appliances into the grid intuitively creates a niche for applying strategic AI techniques typically applied to games for obtaining the best outcome, which is a largely unexplored application due to the novelty of the challenges and the intrinsically multiagent structure of the problem. Nevertheless, it is necessary to prepare frameworks to answer these challenges to mitigate the impact on the existing network infrastructure. This can be done only by importing techniques from relevant specialized fields. The work presented in this project has an aspect of unconventionality in that the use of techniques based on Stochastic Games and dynamic programming in a complex scenario requires to carefully study the structure of the underlying problem, but offers a complete view of the strategic solution found compared to "black box" methods based on function approximation such as deep reinforcement learning, frequently used with complex state spaces. Making the model applicable will require the exploitation of structural peculiarities of the problem and the proposition of methods inspired by the literature on Stochastic Games. The main objectives of the work are as follows:

- Perform a suitable parametric modeling of the flexibility management problem in smart grids in the framework of dynamic programming;
- Study the problem carefully to obtain computational advantages compared to attacking the "raw" state space;
- Propose a solution method to treat the problem in a satisfactory way, that is, a method able to quickly find solutions where dominated strategies are rejected and, vice versa, dominant strategies are retained for a sizeable state space.
- Visualize and analyze the policies resulting from the solutions and propose summary metrics;
- Perform a sensitivity analysis of the summary metrics with respect to the parametrization of the model;
- Simulate agents managing flexibilities with the proposed method together with an electric grid to study the impact of the agents on the underlying grid in terms of, e.g., distribution of voltage.

The next two parts of the work will be dedicated to the exposition of the results and their analysis.

Part II
Results

3 Modeling reality

Generally speaking, the agents steering the flexibilities do so in order to attain a behavior with global characteristics that they find desirable (economic optimality conditioned to user satisfaction), and the choices taken at a certain moment in time influence the underlying state of affairs that will be present in the future.

Furthermore, the agents can be seen as inhabiting an environment constituted by the appliances they manage. They can interact with the environment exerting control actions, such as imparting the decision to charge or discharge a battery, and they can "perceive" measures, or information, about them: the SOC of the battery, whether a set of PV panels is producing energy, etc. The entire set of the measurements the agents can obtain can be seen as an observable state. The evolution of the state of the environment is influenced by their control actions, but also by stochastic phenomena out of their control (the weather, the departure or return of an EV, etc.).

3.1 State and Action Spaces

The choice of a discrete state space is one of the most delicate tasks we are faced with in this work. Big state spaces allow for a more detailed representation of reality, but computationally Stochastic Games suffer from the curse of dimensionality, and therefore the situation becomes rapidly unwieldy as the discretization of some variable becomes finer and as state variables are added. The key objective of the work is to represent and study flexibilities, chiefly EVs, building thermal inertia and stationary batteries, therefore it is natural that most of the variables will be dedicated to capturing these aspects. Another very important aspect for the significance of this study is the distributed generation the agents have as prosumers, typically found in the form of rooftop PV panels. A variable will then be dedicated to representing the production level of the PV panels. Another fundamental variable will have to represent the time of the day, crucially account for events such as EV departure and return or deliberate consumption on the user's part.

Other aspects are important in the definition of the actual state the agent operates in: for example, seasonality will influence the stochastic properties of the state transitions with respect to PV production, for example. However, the strategies studied here aim to respond to the need for a detailed, intra-day appliance management; therefore, factors varying over long timeframes are not included in the state, and will be included, if necessary, as a new instance of the transition probability matrix, giving rise to a new strategy dedicated to the particular macro-condition considered, and interesting comparisons among different macro-conditions might arise.

We now show two possible parametrizations that were considered for the state of a single agent:

It is very important to notice how a part of the defined state, and in particular the *Time of day, PV*

Table 2: Two possible state discretizations of the state space

Variable	Discretization	
	High	**Low**
Time of day	24	12
PV production	3	2
Thermal Buffer	5	3
EV SOC	3	2
EV is in	4	2
BESS SOC	3	2
Total	12960	576
Private	180	24
Public	72	24

production can be considered rather realistically a "public", shared common part of the state when considering more than one agent. The state measured by the single agent h can be written as the concatenation of these two parts.

$$s_{q,h} = (s_{sh}, s_{own,h}) \tag{19}$$

The time of the day is trivially the same for the whole community, whereas weather conditions can be approximated to be equal. Therefore, when considering n agents, one will not need to combinatorially exponentiate the 1-agent state by n, but only the private part. This feature will also be important in Section 4, where it will be leveraged for proposing a solution concept.

The action space will be composed of two true/false independent actions, and therefore of a total of 4 actions:

- $a_0 \rightarrow$ nothing is done;
- $a_1 \rightarrow$ the heat pump system is activated for the period;
- $a_2 \rightarrow$ if possible, the batteries are charged (first the EV, then the stationary battery);
- $a_3 \rightarrow$ both a_1 and a_2 are performed.

3.2 Transition probabilities

In this section we will illustrate how the transition probability matrix P for a single agent is computed. Notice that it is possible to consider and compute the probability transition matrix for a single agent even in the n-agents case because the actions of an agent only influence the private part of its state; this is very important and will be further expanded in Section 5.

3.2.1 Qualitative considerations

It is postulated that the variables that compose the state change according to random events independent of one another; for example, the probability of an EV departing is independent of the probability of the weather changing. Some variables cannot reach all values from the previous values they had: for example, a full thermal inertia cannot be emptied abruptly. Furthermore, some of the variables change in a deterministic way: chiefly the time of the day. Provided that we sort the state vector appropriately (that is, with a hierarchical sort where the time of day is the highest level key), the probability transition matrix will therefore be, qualitatively, in blocks (states where $time = 1$ are always followed by states where $time = 2$) and inside the blocks there will be considerable sparsity due to the reciprocal unreachability of the variable values in a single step. The precise computation of the transition matrix must be done programmatically, considering the joint probabilities of all random events that take place, as is described in the following section.

3.2.2 Evolution rules and exchanged energy

In this subsection, we will illustrate the rules used to generate the probability transition matrix.

- The time of day evolves in a deterministic and cyclic fashion: from 1 to the max discretization value.
- According to the time of the day, there is a tabular probability that the EV will have left in the following period if the EV is currently in, and a probability that the EV will have returned if the EV is currently out.
- When an EV returns, it does so either completely discharged or with one unit of SOC with equal probability. The indicator "EV is in" reads 0 when the EV is not parked, turns to 1 when the EV arrives and then, if the discretization is > 2, it increases 1 unit per timestep until it reaches a maximum value. This indicator can be used to evaluate customer dissatisfaction according to time (the user is dissatisfied with the charge only if the vehicle had enough time to charge).
- According to the time of the day, there is a tabular probability that the user will consume one unit of energy deliberately.
- PV panels produce one or two units of energy per period during the day, according to the state parameter representing the irradiation.
- If the heat pump system is activated, the thermal buffer of the building is increased by a fixed quantity (in the example, 3). If it is not, the thermal buffer is reduced by 1.
- The balance of consumption and production is settled according to the priority rules illustrated in Table 3, that are executed consecutively, in that order. For example, the first line means that the consumption of the user is matched with the production of PV, if any; if there is no PV production, then it consumes SOC from the BESS, if it is charged; finally, if even this source is unavailable, it resorts to absorbing from the grid.

According to the evolution procedure described above, depending on the realization of the random events a new state is reached and a certain quantity of energy $\mathcal{E}(s, a, \theta_i)$ is exchanged with the grid. We indicate with θ_i a combination of realized values of the random events involved (or in other words an element in the space of possible realizations), that we schematize here for convenience:

- EV departure (if EV at home)

Table 3: Consumption and production settling priority.

User consumption	PV $\rightarrow$ BESS $\rightarrow$ grid
Heat pump (if active)	PV $\rightarrow$ grid
EV	PV $\rightarrow$ grid(if charge order issued) $\rightarrow$ BESS
BESS	PV $\rightarrow$ grid(if charge order issued)

- EV return (if EV away)
- EV charge at return (if EV return)
- User absorption

We also define the following function $\mathcal{T}$:

$$\mathcal{T}(s'|s,a,\theta_i) = \begin{cases} 1 \text{ if } s \text{ transitions to } s' \text{ given action } a \text{ and realization } \theta_i \\ 0 \text{ otherwise} \end{cases} \tag{20}$$

The elements of the transition matrix P can then be calculated as follows:

$$P(s'|s,a) = \sum_i \mathcal{T}(s'|s,a,\theta_i)\mathbb{P}(\theta_i) \tag{21}$$

The function $\mathcal{T}$ has to be computed programmatically according to the above rules, while $\mathbb{P}(\theta_i)$ is the joint probability of the realization of the single random events. The EV charge on return has a fixed flat probability on 0 and 1 as mentioned above, while the other events have time-dependent probabilities defined by the curves mentioned above. In the next section we will treat how these curves are chosen.

In the same fashion, the expected value of the energy exchanged with the grid at a state s given action a can be written as $\hat{\mathcal{E}}(s,a)$ calculated as follows:

$$\hat{\mathcal{E}}(s,a) = \sum_i \mathcal{E}(s,a,\theta_i)\mathbb{P}(\theta_i) \tag{22}$$

3.3 Event probability curves

In Section 3.2.2, the mechanism for the evolution of the state was shown. We referenced three tabular, time-dependent probability curves:

- Probability that the EV leaves, if it is currently in;
- Probability that the EV returns, if it is currently out;
- Probability of deliberate consumption of one energy unit by the user.

These curves need to be specified for the model to be complete. We will do so in the following sections.

3.3.1 EV mobility

The EV curves were obtained by elaborating data generated by the mobility simulation tool, MATSim [**?**]. A database with 176806 records was generated, representing mobility requests by the users along a typical day, as modeled by the MATSim. The region of interest is the Canton Ticino, CH. The records include various descriptive fields, among which the most important for generating the probability curve are the following:

- User ID
- Means of transportation
- Departure time
- Travel time
- Departure place
- Arrival place

First of all, the records are filtered, keeping only those in which the means of transportation is by car and either the departure or the arrival place is "home".

Then, a list of all the unique User IDs is generated. The records pertaining to each User ID are then analyzed separately. We therefore end up with a list of all the movements made by the user by car, and we are able to reconstruct the history of movements. This is shown in Figure 1a.

A hourly table is then generated indicating whether the EV is at home connected or not, using the list of car movement events to locate state changes (Figure 1b). The change is assigned to an hour according to either the Departure time or Arrival time indicated in the record, depending on whether the EV is departing or returning home.

From the hourly table of state, a corresponding hourly state transition table is created, with possible events $in \rightarrow out$, $out \rightarrow in$, $out \rightarrow out$, $in \rightarrow in$ (Figure 1c).

We take the rows from the state transition table, and we cumulate them in a series of tables, one per each hour, counting how many events of that kind happen in that hour (Figure 1d). Each User ID adds only one event to each of these tables. The procedure is then repeated for each of the User IDs in the list, so the tables end up populated with many events, forming an aggregated probabilistic model over the sample offered by the starting mobility database.

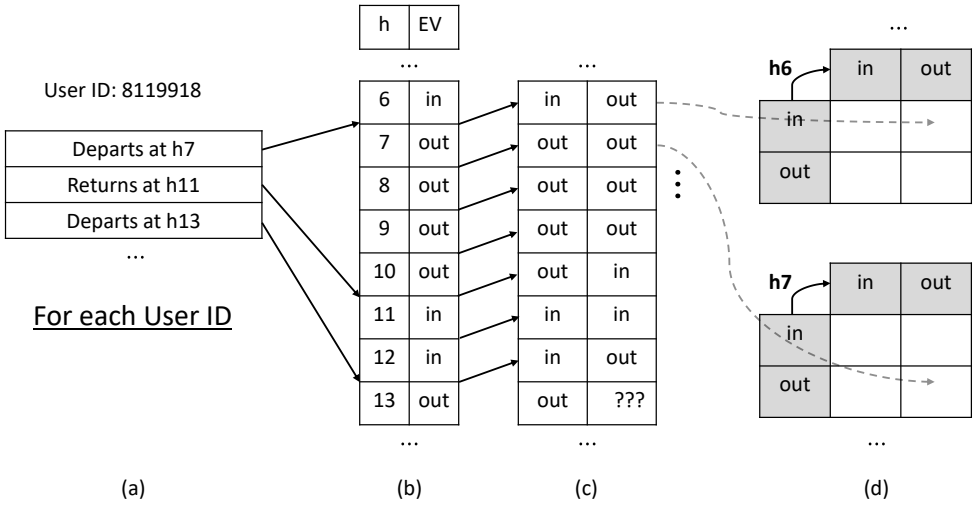


Figure 1: Visualization of the fitting procedure for EV mobility.

Once these tables are created, indicating their elements with $N_{s1\to s2,h}$, for each hour the tabular probabilities are calculated as:

$$P_{dep,h} = \frac{N_{in\to out,h}}{N_{in\to in,h} + N_{in\to out,h}} \tag{23}$$

$$P_{ret,h} = \frac{N_{out\to in,h}}{N_{out\to out,h} + N_{out\to in,h}} \tag{24}$$

The final result of the fitted hourly curves is shown in Figure 2.

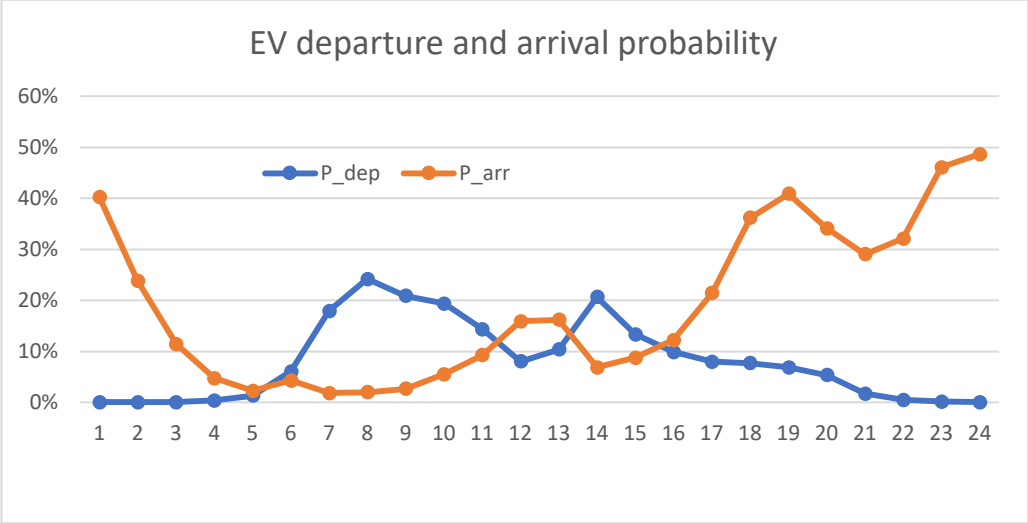


Figure 2: Fitted arrival and departure probability curves.

3.3.2 PV production

Primitive data for solar irradiance were taken from the PVGIS system [33].

The acronym PVGIS stands for Photovoltaic Geographical Information System. The European Commission's Joint Research Center develops and distributes the information system for free. PVGIS is a database that contains irradiance statistics for Europe, Africa, and Southwest Asia. These were measured over an extended period of time so that accurate average values could be derived. These figures of global radiation are an excellent starting point for calculating the potential yield of a photovoltaic system. PVGIS offers a variety of maps containing radiation data that can be downloaded at no cost. The PVGIS system is capable of computing an estimate of the PV yield at a certain latitude and longitude, given an installed peak power on which the data is normalized and a reference yearly time bracket. It is possible to access the PVGIS data and computing capability through a web API, requesting hourly production results.

Once the production data is downloaded and normalized, the data is bucketed in 3 clusters to obtain data compatible with the state discretization. As explained in Section 3.1, the evolution of time and irradiance are studied together, as some value couples are never realized (at night the irradiance is always zero, and in the morning and evening the top level of irradiance is not experienced). The hourly table (Figure 3a) is then straightforwardly converted in a list of transitions between tuples of $(time, PVprod)$ values (Figure 3b), which are then counted (Figure 3c), in a process that bears some resemblance with the one shown for EV mobility.

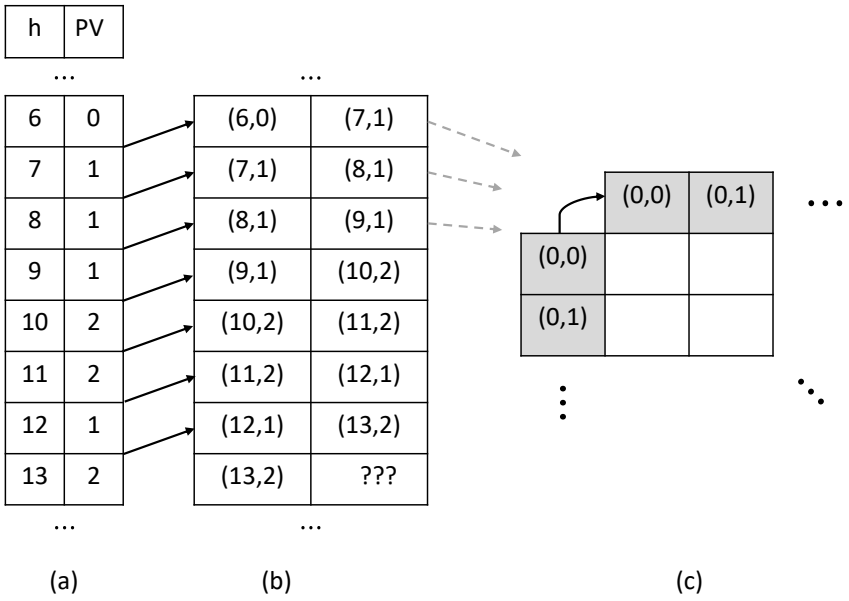


Figure 3: Visualization of the fitting procedure for PV production.

Here we show in Figure 4, to illustrate the final result, the transition table obtained for a fitting performed on the whole year 2016 at coordinates $45°00'00.0''N 8°00'00.0''E$.

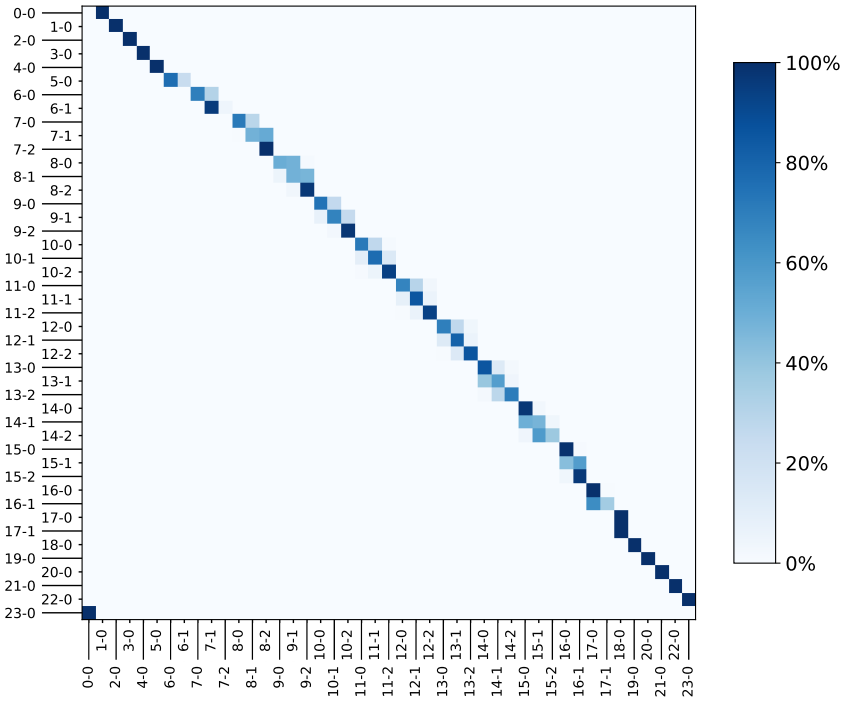


Figure 4: Example PV production fitting result.

Notice that not all the $(time, PVprod)$ combinations may be realized; this allows for the reduction of the theoretical cardinality of the space state, as we will explain briefly.

The use of the same weather data for all the users is justified by the fact that the communities are contiguous; in the case of purely virtual communities, this would be different, but this is out of the scope of the present work, not least because the significance of the tool in offering voltage impact analysis would also be lost. The averaged transition matrix is deemed adequate for generating sensible strategies that the agents could adopt in reality. Furthermore, it would be immediate to generate dedicated or seasonal transition matrices to produce strategies better suited for a particular season; the intended use of the tool in real situations would be to compute many different strategies "offline" and use the most appropriate parametrization from time to time.

3.3.3 Residential load

For the residential load in the standard scenario (cfr. section 8.2), the example probability curve shown in Figure 5 was used, tracing a household consumption profile. The consumption probabilities in the morning and evening hours are high, and this choice was made in order to be able to easily gauge the ability of the strategy to steer the load away from another imposed consumption peak.

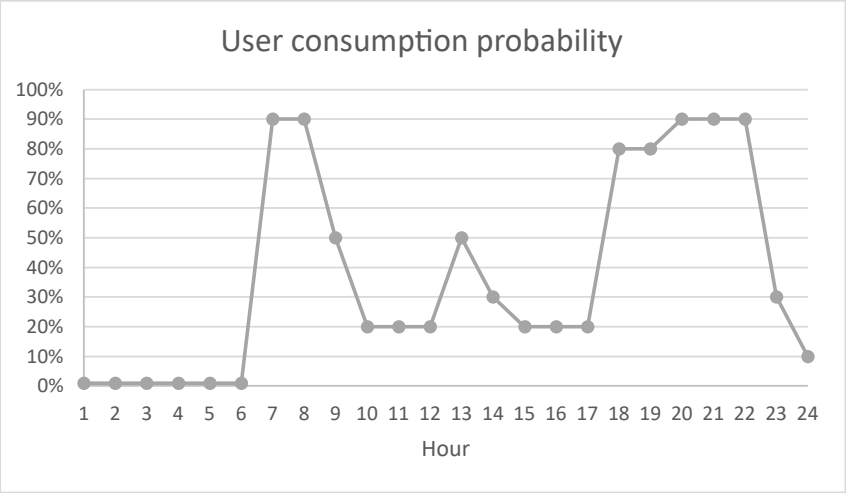


Figure 5: Example user consumption probability curve.

3.4 Example P matrix

An example P matrix generated according to the rules, procedures and probability curves exposed in this Section is shown in Figure 6 for the "big" state space (cfr. Table 2). For simplicity, only the transition matrix for fixed action a_0 is shown. A few important observations:

- The space is reduced from the theoretical cardinality of 12960 to a cardinality of 8100, thanks to the coupling of time and PV production, as described in Subsection 3.3.2.
- The block composition is evident and recalls Figure 4: in fact, that matrix can be interpreted as the transition matrix of the shared part of the state only, and since the rows are sorted by shared part first, the blocks of P precisely retrace the position of nonzero elements in the PV-time fitted matrix. This, naturally, remains true for the transition matrix related to each of the 4 possible actions.
- The visual grid lines separate blocks of states according to their time. The nighttime blocks are smaller, as the states with PV production >0 are completely excluded and are not represented in the rows and columns of the matrix. This corroborates visually the state space reduction.

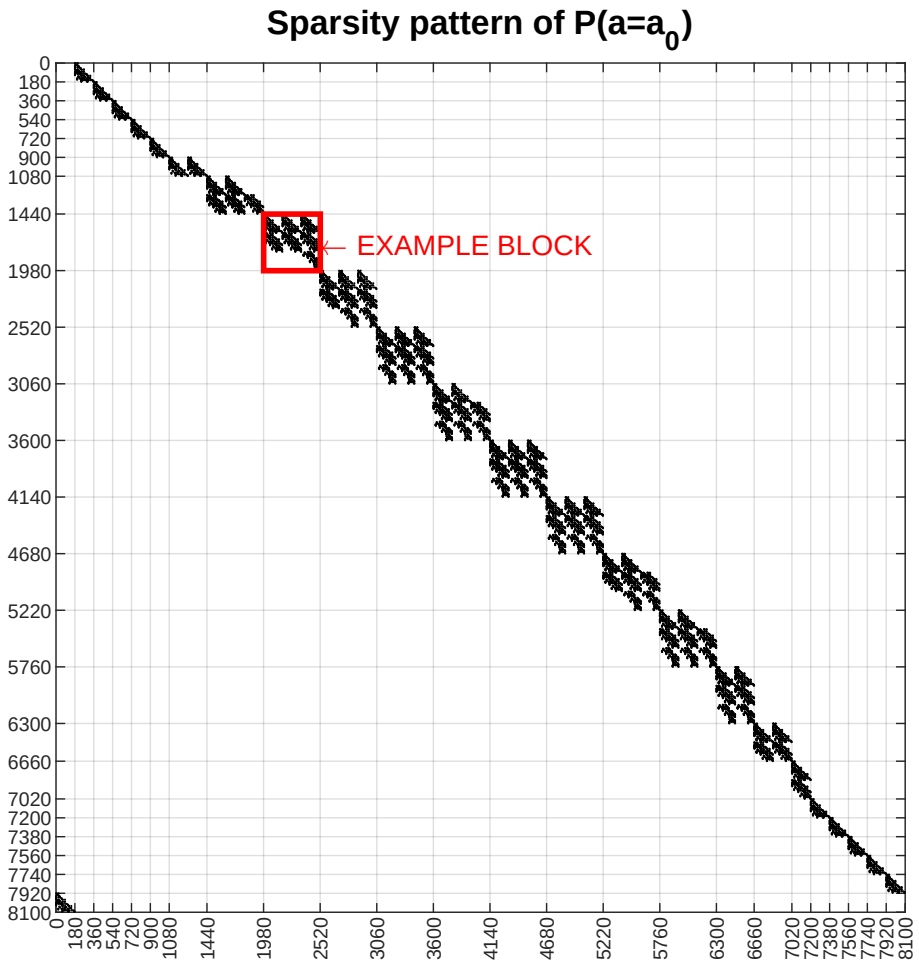


Figure 6: Sparsity pattern for the transition probability matrix.

Figure 7 further shows the intra-block sparsity pattern of the matrix, as not all states are reachable from a certain states in only one timestep, as anticipated in Section 3.2.1. The example block is shown for the transition matrix of each of the actions, showing the fine structure difference of the states reached according to the action taken.

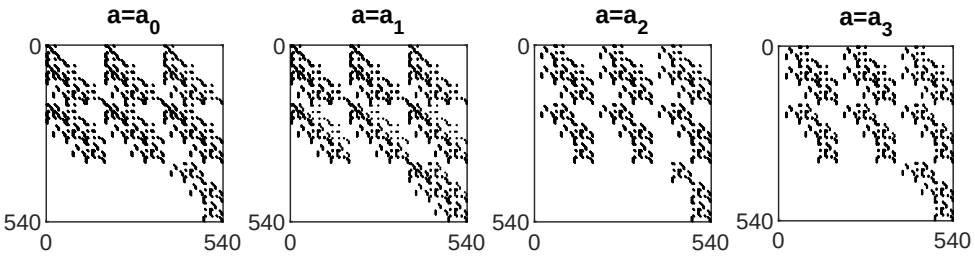


Figure 7: Sparsity pattern for example blocks.

3.5 Rewards

The rewards assigned to the agents are the motor of their choices. The agents are torn between three forces: the dissatisfaction of the customer should he or she experience discomfort (off-range thermal conditions of the building, EV not charged upon the need to use it), the optimization of energy price, and a coupling term accounting for excessive coordination.

The single agent experiences individually the following penalties and prices:

- User dissatisfaction is expressed as a constant negative reward that occurs when an EV is found discharged after it had time to recharge and the user needs it.
- When a state is reached where the thermal inertia of the building reaches 0, the agent receives a constant penalty.
- According to the settling illustrated in Table 3, the agent pays the number of energy units taken from the grid, following a hourly price table similar to the hourly probabilities that parametrize the random events in Section 3.2.2.

Formally, we identify the following quantities:

- The negative reward for the user finding a completely discharged EV upon wanting to use it, R_{tot}.
- The negative reward for each unit of discharge when the user finds an EV partially discharged (after having given it enough time to recharge), which we will call R_{part}
- The hourly negative reward for letting the building thermal inertia reach 0, which we will call R_{therm}.

Quantitatively, the rewards have to be regarded as a multiple of a fixed cost, corresponding here to the cost of an energy unit. This whole computation is therefore adimensionalized with respect to the cost of the energy unit.

As far as the coordination term is concerned, each absorbed energy $\mathcal{E}$ will contribute to the collective effect. Disruptive effects on the grid are nonlinear, so the absorbed energies are aggregated with the square-sum. Therefore, at each step and with random event realization θ, the agents will receive an additional penalty term of the following form, modulated through a coefficient C:

$$R_{collective}(s, \mathbf{a}) = -C\left(\sum_i \mathcal{E}(s, a_i, \theta)\right)^2 \tag{25}$$

3.6 Final remarks

The type of modeling presented in this section is interesting because, once an algorithm to find sensible strategies in a reasonable time is in place, the reward parameters can be tuned for the purpose of realizing many kinds of sensitivity analyses, allowing for the quantification of several effects that the existence and operation of an energy community such as the one modeled here can have.

In particular, it will be possible to see what is the effect on available steering power for the community if the "flexibility" of the end user is increased – that is, if the dissatisfaction penalties are attenuated. This result is of particular importance because it helps in answering a hot research question: how

can "flexibility" be quantified? The economic advantage given to users and the effectiveness of the community as a service provider can also be explored when varying the probabilistic model of Section 3.2.2, enabling insights on the potential of this kind of solution according to the underlying situation. Finally, when the trained agents will be co-simulated together with various configurations of electrical distribution networks, the effects of the agents behavior on relevant grid parameters, such as voltage distribution, can be evaluated according to the variation of the probabilistic model and the coordination penalty.

4 Structural peculiarities

As anticipated previously, the specific problem of finding good strategies for the agents under the model exposed in Section 3 has a few peculiarities that hopefully can be leveraged in the pursuit of a method capable of computing good agent strategies under a sizeable state space.

4.1 States and Rewards

The single-agent state space is formed by the parameters indicated in Table 2. One important observation was already made, that is, part of the state can be considered as *shared* (Time of day, PV production, PV forecast) and will be shared among all agents, whereas the rest of the state pertains only the private parameters. We further illustrate this state of affairs in Figure 8, introducing notation that will be useful later.

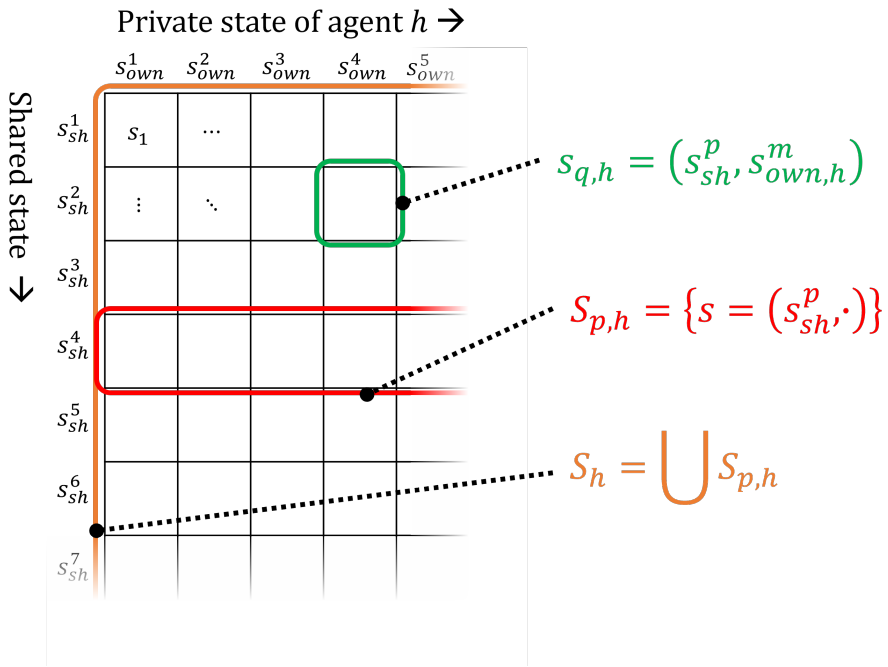


Figure 8: Subsets of the state space of one agent.

- $s_{q,h}$ is an individual state numbered as q for agent h, comprising a shared part numbered with p and a private part numbered with m;
- $S_{p,h}$ is the set of all individual states for agent h where the shared state corresponds to the particular shared state numbered with p;
- S_h is the whole single-agent state space of agent h.

Notice once again that this is the state space measured by a single agent partaking in the Stochastic Game; a state of the whole Stochastic Game s, belonging to the global state space S as introduced in Section 1.2, has the following form:

$$s = \left(s_{sh}, s_{own,1}, \ldots, s_{own,n}\right) \tag{26}$$

A relevant feature of our problem is that the actions taken by the agents only influence the private part of the state, and the only state on which the reward to each agent depend is its own. Only the part of the reward concerning the coordination is influenced by the other agents, and only through the next action, not directly by the other agents' state. Importantly, the evolution of the state for the single agent only depends on its actions and on natural chance. In the light of these considerations, one can view our problem as the union of an MDP played by the agent for managing its appliances and a coordination game played at each step as a collateral effect of the actions taken. Conversely, if the penalty for the coordination game was reduced, the solution of the whole game would simply be the solution of the MDP of each agent. A good initialization strategy for the agents is, therefore, the solution of their "private" MDP.

4.2 Imperfect information and agent symmetry

Until now we reasoned about the global state space as the combination of the "shared" part and as many "private" parts as there are agents. For a significant number of agents, this would translate very rapidly in a global state space of unwieldy size, even using the reduced version of the agent state space. As we already observed, though, in reality the agent on the field will *not* have access to the other agents' private states, and therefore would only have the possibility to play some kind of "blend" of a global-state-specific strategies according to its estimate of the distribution of the states of the other agents, which would be in itself nontrivial to elaborate. A more natural way of dealing with this is incorporating the *imperfect information* nature of the game directly. Albeit, normally, imperfect information games are more difficult to solve, in our case we can exploit this characteristic to find an efficient way of finding strategies solving the game from the perspective of a single agent. During play, the single agent can consider its own state as an *information set*: that is, the knowledge of being in a certain state limits the possible global states the whole system can be in, but there are several. This is a very important, yet subtle, conceptual passage that is worthy of reiteration. If solving the game with the knowledge of the global state would lead to strategies custom-tailored to each single combination of the joint agents' states and therefore more optimal, this strategy would not be deployable directly in reality. Under the "natural" imperfect information condition the agent has to be able to play only according to the knowledge of its own state, as it is the only state it has actual access to. Therefore, we choose to solve the game per single-agent-state, that is, per each single agent information set. The resulting strategy will be "coarser" – that is, the agent will answer in the same way to a whole set of global states: the information set induced by its own state (cfr. Figure 9). Nevertheless, this has two crucial advantages: it corresponds directly to a real implementation and allows us to consider a dramatically smaller state space, which is

providential for the tractability of the problem, and by requiring to consider just one agent will afford us the use of the bigger, more interesting version of the state space for that single agent.

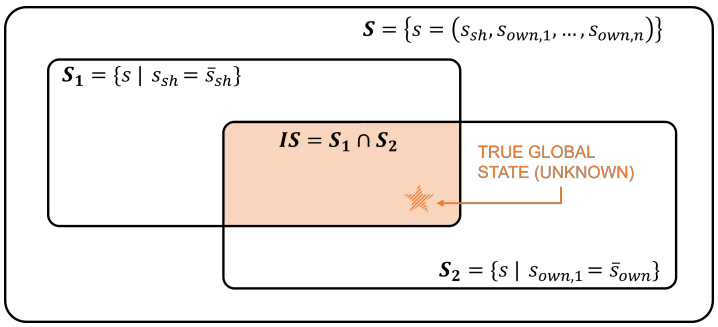


Figure 9: Illustration of the global state space and the agent's information set IS.

Generally speaking, every agent could have its own private transition probabilities, rewards and state space. To further aid in finding a solution method, we choose, at least initially, to leverage "symmetry" between the agents. If the agents have the same state, transition probabilities and rewards, it can be said that they are indistinguishable and therefore should have, in the end, the same strategy.

$$S_{p,h'} = S_{p,h} \quad \forall p, h, h' \tag{27}$$

$$x^{h}_{s,a} = x^{h'}_{s,a} \quad \forall s \in S_p, a, h, h' \tag{28}$$

In an iterative algorithm such as the one sketched in the following section, this property is important, because at each iteration we can leverage the hypothesis that the other agents respond with the same strategy of the agent from whose perspective we are solving from, allowing the algorithm to gauge the effect that an increase in the frequency of an action has on coordination directly, optimizing it in one shot. In a second phase, once this is done, this hypothesis might be relaxed allowing to have population of several "types" of agents; in this case, the problem will arise if it is better to optimize all populations simultaneously or one at a time, but this will be dealt with according to how the project unfolds.

5 Two-level approximation methodology

5.1 Introduction

As we reviewed in Section 1.1, solving an MDP with a known model (P and R matrices) is a matter of iteratively evaluating a "Bellman Optimality Operator" to find a value for the states, equal to the discounted cumulative rewards obtained starting from that state and playing under an optimal policy; and an optimal policy is just a greedy policy over those values. From the perspective of an agent living in a population of similar agents, this line of reasoning is still partially applicable in Stochastic Games. First notice that, if all the agents but one use a fixed policy, the game simplifies to an MDP for the only agent that is allowed to change policy, as the other agents become in a certain sense "part of the environment", which is stationary. But in reality, the other agents experiment exactly the same incentive to change their behavior.

Therefore P and R for the agent are not stationary under a change of policy, and in fact, to mantain the same spirit of the MDP solution they should evolve in the way dictated by the same, simultaneous update for all the agents. As Buşoniu et. al. [7] observe:

> *Nonstationarity of the multiagent learning problem arises because all the agents in the system are learning simultaneously. Each agent is, therefore, faced with a moving-target learning problem: the best policy changes as the other agents' policies change.*

This requires a second step: after having found a strategy that is optimal over the values induced by the strategy of the previous iteration, the previous strategy is not "valid" anymore and we must update these values according to the new strategies, in a loop. Under the particular structure of the problem described in the previous Section – that is, considering that the transition of the state of the agent is not influenced by the joint action, but only by her own – the algorithm has experimentally performed well, reaching at least a subset of sensible strategies. As already mentioned, it is difficult to find solution guarantees for general-sum Stochastic Games, but the additional structure might be useful.

5.2 Description

In this section we propose a two-step method, similar to Policy Iteration, to take care of this more complex situation, inspired by the methods proposed by Ganzfried and Sandholm to solve approximate equilibria (equilibria restricted to just two actions for all players, "jam" or "fold") in multiplayer tournament no-limit Texas hold'em Poker, which they model as a Stochastic Game [34]. In their work, the authors need to leverage Fictitious Play [35] for the purpose of finding a policy suitable for the current estimate of the state values, since they are solving an opportunely modeled extensive form of the game in which the players do not have symmetry at each state, but are distinct strategically and in terms of state space by the order in which they play. Once they get the updated strategy, they run some variant of MDP method to find an update for the state values, and the procedure is repeated iteratively.

In our case, instead of fictitious play, we attempt to use a generalized version of the Bellman Optimality Operator with the following characteristics:

- Instead of picking the pure action that maximizes reward + value of the next state, the maximization takes into account the term described in 3.5, potentially generating mixed strategies;
- Instead of performing the maximization at each state, the vector of all actions for all states over the entire $S_{p,h}$ set (Cfr. Figure 8) is optimized monolithically for each s^p_{sh}.

The reason for the second point is the following. Since at each observed agent state $s_{q,h}$ comprising a certain public part s^p_{sh}, the distribution of the states of the other agents spans the whole $S_{p,h}$ set, the choices at each single agent state in $S_{p,h}$ are relevant at the same time for the expected value of the collective term. At each state $s_{q,h}$, though, the agent can infer a different distribution of the other agents' state, so the expected collective term must be calculated by weigh-summing over $s'_{q,h}$ the effect of the action profile taken in state $s'_{q,h}$ for the probability of the other agent's state $s_{q,h'}$ being $s'_{q,h}$, given that the observable agent is in $s_{q,h}$. The calculation of this probability, $\mathbb{P}(s_{q,h'} = s'_{q,h} | s_{q,h})$, is nontrivial and is covered in Section 6. Solving the $S_{p,h}$ subsets monolithically is also an advantage in terms of computation time, as cycling through states and solving a maximization problem for each would be more time-consuming even if the problem would be smaller.

Recalling from Section 3.5 the expected exchanged energy function $\hat{\mathcal{E}}$, and calling $x = [x_{s,a}]$ the optimization variable, that is the strategy matrix containing the probabilities of performing action a in state s for each s included in the considered $S_{p,h}$, we maximize the following convex formulation:

$$\begin{aligned} \max_{x} \quad & \sum_{s \in S_{p,h}} \Big(\sum_{a \in A} Q(s,a) x_{s,a} - c \sum_{s' \in S_{p,h}} \mathbb{P}(s'|s) \big(\sum_{a \in A} x_{s',a} \hat{\mathcal{E}}(s',a) \big)^2 \Big) \\ \text{s.t.} \quad & x \in \mathbb{R}_{+}^{|S_{p,h}| \times |A|} \\ & \sum_{a \in A} x_{s,a} = 1 \ \ \forall s \in S_{p,h} \end{aligned} \tag{29}$$

The overall procedure is summarized in the following pseudocode.

Algorithm 3 Multiagent Symmetric Policy Iteration

procedure TMSGSYMMETRICPOLICYITERATION(S_h, A, P, R, γ, C, τ)
  Initialize $x(s), x'(s)$ randomly.
  $niter \leftarrow 0$ **while** x *is not converged* & $niter < \tau$ **do**
    Find V^x using Equation (2) **for** $S_{p,h} \in S_h$ **do**
      $[x'(s) \forall s \in S_{p,h}] \leftarrow$ solution to Problem (29)
    $x(s) \leftarrow x'(s)$
    $niter \leftarrow niter + 1$
  return x

An interesting observation is that the method reverts to being vanilla Policy Iteration when the coefficient of the collective term $C = 0$.

The method was tested on example games with about 10^5 states and was found able to reach in a few minutes convergence or, at least, a cyclical repetition of solutions, and therefore is set to be the "mainsail" for the quantitative part of the work.

6 Inference of the distribution of Private States

In the light of Equation (29), we want to tackle the problem of conditional state distribution from the perspective of an agent. Given a joint policy x, in fact, the knowledge of an agent of being in a certain state has an effect on the distribution of states of the other agents. This occurs not only because the other agents must be in a state with the same s_{sh}, but also because the private part of the state is indicative of what has happened recently, which in turn has an effect on the most probable states the other agents could be in. We will now present two methods of doing so. The first one was the first to be developed, but the second is more precise and fast and it was the one routinely used in the calculations in this work.

6.1 First method: path-based

First of all, let us observe that, fixed the joint policy $\bar{x}$, the Stochastic Game becomes a Markov Chain with probability transition matrix $P_{\bar{x}}$ (cfr. the concept of *on-policy state distribution*, for example in [3]). The on-policy probability matrix is fixed: $\hat{P}(s'|s, \bar{\mathbf{x}}) = P_{\bar{\mathbf{x}}}(s'|s)$. The long-term distribution of states $D_{\bar{x}}$ is easily computable, for example by finding the biggest eigenvector of $P_{\bar{x}}$ with power iteration [36]. Therefore, it is possible to reconstruct the approximate probability of all the possible state "paths" $\boldsymbol{\sigma}^i = [s_{-l+1}, ..., s_{-1}, s_0 = \sigma]$ of a certain length l that led the agent to be in the current state σ:

$$\tilde{\mathbb{P}}(\boldsymbol{\sigma}^i) = D_{\bar{x}}(s_{-l+1}) \prod_{j=-l+2}^{0} P_{\bar{\mathbf{x}}}(s_j|s_{j-1}) \tag{30}$$

Each one of these paths corresponds to a sequence of $s_{sh,i}$ that, in that path, were valid also for the other agents. These s_{sh} can be observed, from the point of view of the agent, as *emitted symbols* of the other agents' actual path, conveying partial information about their complete state. Therefore, the problem of inferring the state probability distribution for the other agents reduces to inferring the probability distribution of the final state from the observation of the symbols, which is exactly what is done in *Hidden Markov Models* (HMMs) and is accomplished with the *Forward Algorithm* [37], known also with the name of "filtering". In generic Hidden Markov Models, there can be a probability of emitting a certain symbol at a certain state, but in our case the symbol corresponding to s_{sh} is always emitted with certainty. The final distribution is then the weighted sum of all the path-distributions. Formally, if we call $\boldsymbol{\Gamma}(\sigma)$ the vector of state probabilities for another agent given that the observing agent is in state σ, that is $\boldsymbol{\Gamma}(\sigma) = [\mathbb{P}(s'|\sigma)\forall s']$, and with $\tilde{\boldsymbol{\Gamma}}(\sigma)$ its approximation:

$$\tilde{\Gamma}_{\boldsymbol{\sigma}^i} = [\tilde{\mathbb{P}}(s_j|\boldsymbol{\sigma}^i_{sh})\forall j] \tag{31}$$

$$\tilde{\boldsymbol{\Gamma}}(\sigma) = \sum_i \tilde{\mathbb{P}}(\boldsymbol{\sigma}^i) \cdot \tilde{\Gamma}_{\boldsymbol{\sigma}^i} \tag{32}$$

Longer lengths l of "simulated observed" sequences result in greater accuracy of the approximation, and in particular the true probability would be obtained for $l \to \infty$. The computational requirements, though, scale combinatorially with the length of the sequences. Fortunately, with a sparse probability transition matrix, the enumeration of the possible paths is drastically reduced and furthermore even short sequences ($l \approx 3$) are enough to get a reasonable accuracy.

6.2 Second method: joint Markov process

In this section we will describe a second, different way of obtaining $\mathbf{\Gamma}(\sigma)$. The idea is to consider the two agents a_1 and a_2, and compose a suitable "joint" probability transition matrix P_j for the Markov process $\mathcal{M}_j$ representing the joint evolution of the two agents, including the constraint of being in the same state subspace at all times. The states of this joint process are 2-tuples (s_m, s_o) composed of pairs of states of the original process $\mathcal{M}$. The stationary distribution of the states of this joint process is therefore the long-term probability of finding the agents in a certain pair of states, from which as we will see $\mathbf{\Gamma}(\sigma)$ is easily derivable for each state $\sigma \in S$. While relatively straightforward, this method involves a matrix of dimension $n^2 \times n^2$ the sparsity and redundancy of which must be entirely exploited, and several aspects of the implementation discussed here are important to keep the memory usage under control and get acceptable runtimes.

6.2.1 Construction of P_j

Consider the situation where a_1 experiences the state transition $s_m \to s'_m$. The probability of this transition is given by definition by the corresponding element of P. The agent a_2 must, at the same time, experience a transition $s_o \to s'_o$ where $\mathcal{S}(s_m) = \mathcal{S}(s_o)$ and $\mathcal{S}(s'_m) = \mathcal{S}(s'_o)$, which means that it must have started from some state in the same subset as s_m and must have ended in the same subset as s'_m. Therefore, all the possible transitions experienced by the second agent are in the block $P[\mathcal{S}(s_m), \mathcal{S}(s'_m)]$, which becomes in a certain sense the probability transition matrix for a_2. Since the possible final states are limited at $\mathcal{S}(s'_m)$, the rows of the block must be normalized to sum 1, i.e., the resulting matrix needs to be right stochastic, as every probability transition matrix is. In other words, the matrix of the same size as P composed of all zeros except for elements with row in $\mathcal{S}(s_m)$ and columns in $\mathcal{S}(s'_m)$ copied from P and row-normalized to 1 is the probability matrix for a_2 conditional to a_1 having experienced transition $s_m \to s'_m$. An illustration of this construction is given in Figure 10. We will informally call a matrix constructed in this way a *conditional block*, for brevity.

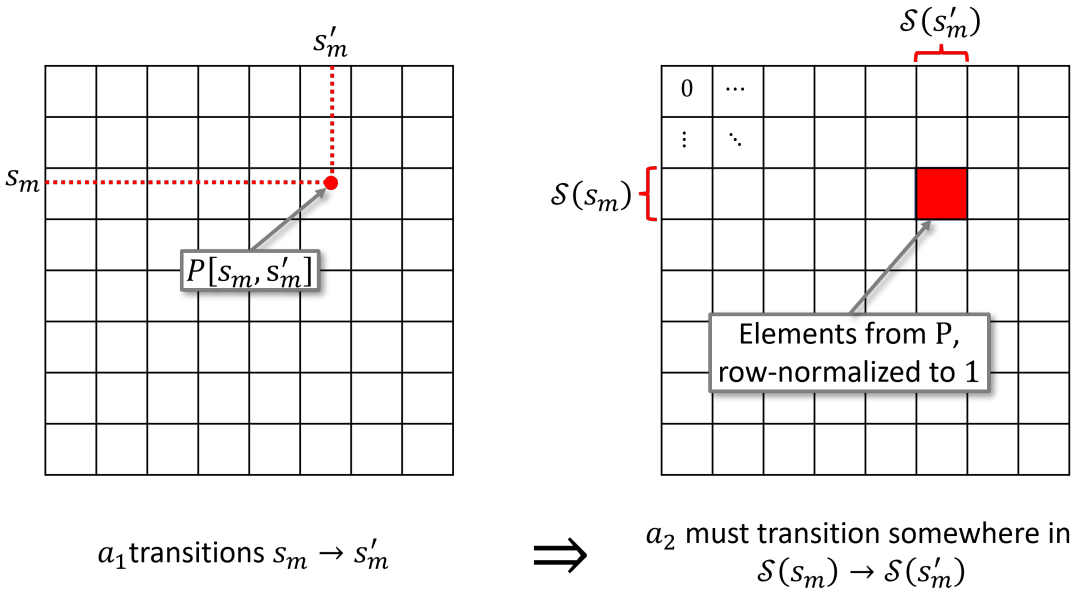


Figure 10: Construction of a *conditional block*.

Applying this process for each possible transition experienced by a_1, we can obtain the whole joint probability transition matrix for couples of states. The construction procedure is shown in Algorithm 1. The `Map` function creates an instance of an *associative array data type*, whose lookup and insertion operations are indicized by angular brackets. For brevity, we allow an array to be indicized with a sequence $a : b$, meaning all elements from index a to index b.

Algorithm 1: Construction of P_j (in COO sparse representation)

Data: P

begin
- $N \leftarrow$ `Order(`P`)`
- $blocks \leftarrow$ `Map()`
- $total \leftarrow 0$
- **for** *each nonzero element* p_{s_m,s'_m} *of* P **do**
  - **if** $\left(\mathcal{S}(s_m), \mathcal{S}(s'_m)\right)$ *already in* $blocks$ **then**
    - $total = total +$ `Length(`$blocks\langle\mathcal{S}(s_m), \mathcal{S}(s'_m)\rangle$`)`
    - **continue**
  - $P'_{data}, P'_{row}, P'_{col} \leftarrow$ `MakeBlock(`$P, \mathcal{S}(s_m), \mathcal{S}(s'_m)$`)`
  - $P''_{data}, P''_{row}, P''_{col} \leftarrow$ `NormalizeRows(`$P'_{data}, P'_{row}, P'_{col}$`)`
  - $blocks\langle\mathcal{S}(s_m), \mathcal{S}(s'_m)\rangle \leftarrow P''_{data}, P''_{row}, P''_{col}$
  - $total = total +$ `Length(`P'_{data}`)`
- $P_j^{data} \leftarrow$ `Array(`$total$`)`
- $P_j^{row} \leftarrow$ `Array(`$total$`)`
- $P_j^{col} \leftarrow$ `Array(`$total$`)`
- $c \leftarrow 0$
- **for** *each nonzero element* p_{s_m,s'_m} *of* P **do**
  - $P''_{data}, P''_{row}, P''_{col} \leftarrow blocks\langle\mathcal{S}(s_m), \mathcal{S}(s'_m)\rangle$
  - $nel \leftarrow$ `Length(`P''_{data}`)`
  - $P_j^{data}[c : c + nel] = p_{s_m,s'_m} \cdot P''_{data}$
  - $P_j^{row}[c : c + nel] = P''_{row} + N \cdot s_m$
  - $P_j^{col}[c : c + nel] = P''_{col} + N \cdot s'_m$
  - $c \leftarrow c + nel$
- **return** `COO` *matrix of shape* $N^2 \times N^2$ *with data* $\left[P_j^{data}, P_j^{row}, P_j^{col}\right]$

function `MakeBlock(`P, S_i, S_j`)`
- $P'_{data} \leftarrow$ `List()`
- $P'_{row} \leftarrow$ `List()`
- $P'_{col} \leftarrow$ `List()`
- **for** $s \in S_i$ **do**
  - **for** $s' \in S_j$ **do**
    - **if** $p_{s,s'} \neq 0$ **then**
      - `Append(`P'_{data} , $p_{s,s'}$`)`
      - `Append(`P'_{row} , s`)`
      - `Append(`P'_{col} , s'`)`
- **return** $P'_{data}, P'_{row}, P'_{col}$

Before discussing the algorithm itself, let us comment an important implementation choice. Matrices P', P'' and P_j are represented throughout the algorithm as a triplet of ordered arrays of equal length labeled $data, row, col$. In this sparse representation, each nonzero element of the matrix is identified by a value, a row-index and a column-index found at the same index inside these arrays. This representation has a few important advantages that aid in keeping memory and time usage as low as possible:

- Matrix multiplication by a constant (Line 20) is fully vectorized;
- Offsetting the indices of the whole matrix by a constant (Lines 21 and 22) is fully vectorized;
- Creating a matrix by joining its parts together (the assignment part in the aforementioned lines) is a matter of sliced array assignment;
- It is immediate to translate this format in the standard *COO* (coordinate) format [38] (which is compatible with ARPACK, that we will mention briefly).

In the algorithm, the first top-level `for` cycle generates all the conditional blocks in an economic way, i.e., the S_i, S_j block is generated only if there exists a state transition in P from a state in S_i to a state in S_j. The actual generation procedure as in Figure 10 is done in the function `MakeBlock`. The blocks are stored in a key-value map `blocks`, and the cumulative `total` counter stores how many elements the final matrix P_j will have. At Lines 13-15 the arrays representing P_j are pre-allocated in memory to favor speed during the assignment. In the second top-level `for` cycle, one a_2 block for each possible transition experienced by a_1 is "tiled" into P_j. Before adding it, the block is first multiplied by p_{s_m,s'_m} to obtain the probabilities of the joint transition (Line 20). In fact, by probability rule $P(A \cap B) = P(A) \cdot P(B|A)$ the probability of the entire block of joint events $(s_m, s_o) \to (s'_m, s'_o)$ is $p_{s_m,s'_m} \cdot P''$, as all individual conditional transition probabilities in P'' have to be multiplied by p_{s_m,s'_m}.

6.2.2 P_j structure and stationary distribution

Once P_j is assembled, we turn to the task of computing the stationary distribution of $\mathcal{M}_j$. As explained in the Appendix, the stationary distribution can be found by searching for the largest eigenvalue-eigenvector pair of $P_j^\intercal$, but the structure and size of P_j are worthy of a dedicated comment. P_j, at least with a choice of reasonably balanced subsets, is inherently sparse, as by construction is composed of tiles of size $n \times n$ where only a rectangle between two subsets may contain nonzero values. If P is in turn sparse, as it is in the majority of applications and in the motivating example presented here, the sparsity of P_j is further enhanced. It is not easy to derive a quantitative estimate for the sparsity of P_j in the general case, since it strongly depends on the particular choice of S_i and structure of P. For k subsets of roughly equal size, though, the following relationship holds:

$$dens(P_j) \approx \frac{dens(P)^2}{k^2} \tag{33}$$

Where $dens(A) \to [0, 1]$ is the matrix density measure consisting in the ratio between the number of nonzero elements and the total number of elements in A. Both lower density (higher sparsity) of P and the granularity of the subdivision in subsets have a quadratic effect on reducing the density (increasing the sparsity) of P_j, and therefore by using the sparse representation this method remains usable for sparse P and a high k even if the number of elements of P_j is $n^2 \times n^2$. The P matrix presented in Figure 6 has a density of 7.44×10^{-4} and with 96 possible public substates this translates into a density of 5.99×10^{-11}. The computation of the stationary distribution in this paper was carried out with ARPACK [39], a

high-quality implementation of Arnoldi iteration methods for finding eigenvalue-eigenvector pairs in large sparse matrices. As explained in the Appendix, only the pair of largest magnitude is necessary, and ARPACK can be configured to look directly for it. In principle, P_j could be passed directly to ARPACK, but an important optimization step can be made. As mentioned above, the states of $\mathcal{M}_j$ are pairs (s_{a_1}, s_{a_2}). By construction, the entire cartesian product $S \times S$ is used as state space for $\mathcal{M}_j$. Yet in reality many of those pairs are known to be impossible, since they are composed of two states belonging to different subsets. This means that those rows and columns of P_j are known to be all zeros, and the corresponding elements of the eigenvector are not important; nevertheless, passing P_j as is increases the computational burden on ARPACK. Therefore, in the implementation, we keep track of the indexes of P_j that belong to the same subset and therefore could have nonzero elements, and before passing P_j to ARPACK it is "sliced" with it.

Notice that by construction, the sorting of the indices (s_{a_1}, s_{a_2}) results by cycling hierarchically all s_{a_2} for each s_{a_1}. Therefore, the integer corresponding to the row/column of joint state (s_{a_1}, s_{a_2}) is $s_{a_1} \cdot n + s_{a_2}$. If the S_i subsets are stored in numeric arrays, the "lean" version of P_j can be calculated in the following vectorized way (Algorithm 2). For notation brevity, we allow here to square-bracket indicize a 2D matrix with two arrays, which means the submatrix obtained by deleting the rows whose indexes are not in the first array and the columns whose indexes are not in the second array. Furthermore, we allow to sum a scalar and an array, meaning that the scalar is summed to each element of the array.

Algorithm 2: Slicing of P_j

Data: P_j, S_i, n

begin
- $nzi \leftarrow$ `Array ()`
- **for** $i \in 1...k$ **do**
  - `Cat` $(nzi, i \cdot n + S_i)$
- $P_{j,lean} \leftarrow P_j[nzi, nzi]$
- **return** $P_{j,lean}, nzi$

After the eigenvector is found, the version of the stationary distribution for $\mathcal{M}_j$ over the entire $S \times S$ space, π_j, is recovered by inserting the elements of the eigenvector of $P_{j,lean}$ in a vector of zeros of length n^2, at the indices provided by the nzi slicing index from Algorithm 2. Once the stationary distribution is found, it is easy to recover $\mathbf{\Gamma}(\sigma)$ for each state σ. In particular, it suffices to reshape π_j in an $n \times n$ matrix, where each row m contains the states $(s_m, \cdot)$. By how the sorting is constructed, this reshaping can be performed directly by dividing π_j in consecutive chunks of length n and stacking them in order. If the rows of the obtained matrix are normalized to 1, they become $\mathbf{\Gamma}(\sigma_m)$ for each row m.

Part III

Analysis and discussion

7 Training strategies

7.1 Convergence

In this section, we will analyze empirically the convergence properties of the method presented in Section 5, which is used to train strategies throughout the work. The analysis in this section is carried out without deliberate user consumption (or, alternatively, with an user consumption curve always equal to zero), in order to "minimize the moving parts" and provide a reference scenario that lends itself to easier interpretation. This is equivalent to the scenario that will be described in Section 8.1.

The training is conducted with a maximum number of iterations of 15. The convergence measure chosen for evaluating the performance is the following:

$$\mathrm{mean}(|V^x_{t+1} - V^x_t|) = \mathrm{mean}(|\Delta V^x|) \tag{34}$$

Which gives an indication of the "stability" of the values assigned to the states across two consecutive iterations.

The training is repeated for several reward setups, in order to see the effect on convergence of the different incentive structure. Intuitively, we vary the objective structure the agents are trying to maximize to see how much it triggers nonstationary effects. The varied quantities are:

- The C coefficient (see Eqq. 25 and 29)
- R_{tot}
- R_{part}
- R_{therm} (see Sec. 3.5)

In Figure 11, we show convergence results for 32 training episodes. The evolution of $\mathrm{mean}(|\Delta V^x|)$ along the iteration is shown for each of them. The results are repeated 4 times, with different color-coding according to the value of one varied quantity at a time. The iterations were stopped if $\mathrm{mean}(|\Delta V^x|)$ fell under 1%.

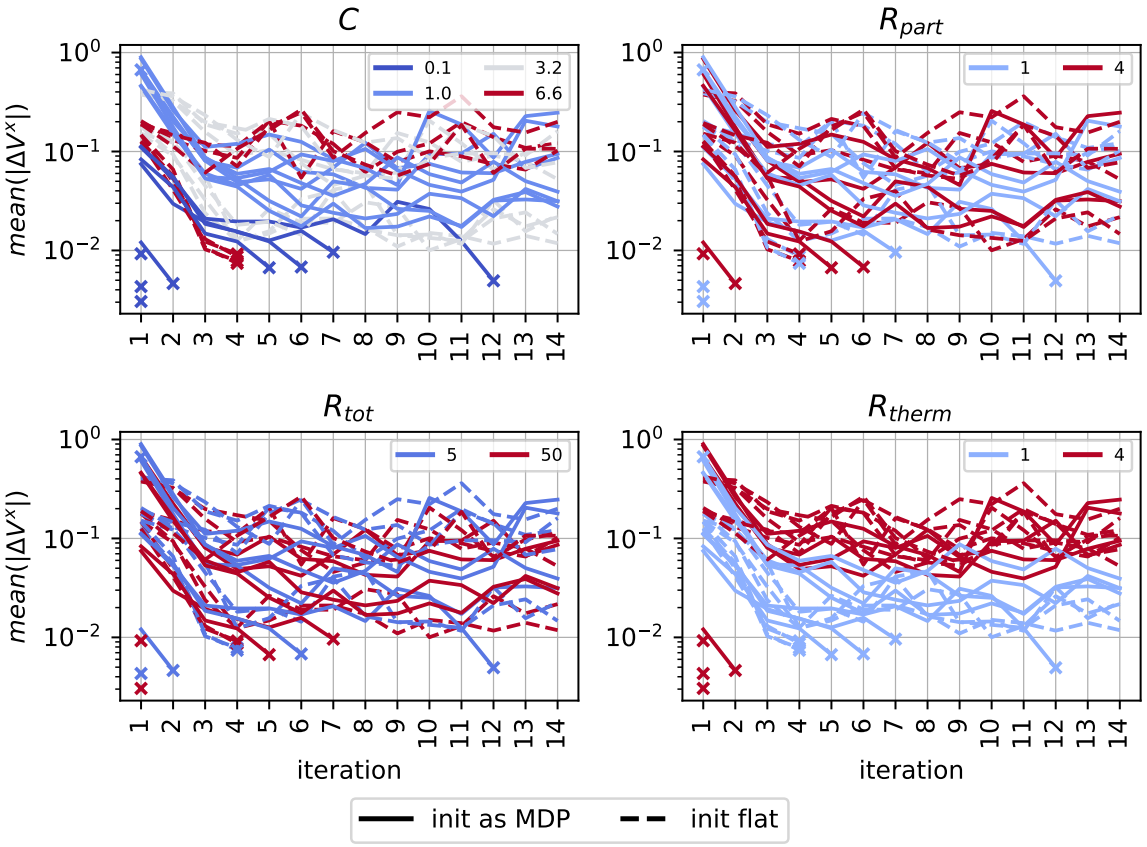


Figure 11: Convergence measurements.

A few words should be spent on the initialization of the initial strategy on which the values are computed. As we observed, for $C = 0$ the problem becomes a regular MDP. For low values of C, and more precisely for $0.0 < C < 3.0$, we initialize the strategies with the strategy for the corresponding MDP with $C = 0$, which can be found with regular policy iteration. For $C > 3.0$, the initial strategy is flat (all action equiprobable in all states). The logic is that for $C \to \infty$, the correct solution so for higher C values the strategy should be more distributed.

Some very interesting results emerge from this figure. Those episodes for which the iterations stopped early are identified by a $\times$ symbol on the last iteration. Almost all of the episodes that converged early (apart from four that we will analyze later) are characterized by the lowest value of the C coefficient. This can be explained rather intuitively: as we saw, the lower the value of C, the more similar to a regular MDP the problem is; therefore the solution is found after "adjusting" the original solution.

Notice now the R_{therm} box. This parameter seems to differentiate quite noticeably two clusters. With high R_{therm}, the episodes that do not converge early tend to stabilize around a higher value of $\mathrm{mean}(|\Delta V^x|)$. One interpretation could be that, being the thermal action heavily time-shiftable due to the inertia, with harsher penalties the iterations tend to move those actions more.

We notice now that the four high-C episodes that stop early are all characterized by a low R_{therm}. With high R_{therm}, instead, the high-C episodes are the ones that present among the highest instability. This

is an interesting phenomenon, and it suggests once more the importance of R_{therm}, as if there was a "tipping point" below which high C stabilizes the strategy and above which it does the opposite.

The other two parameters, R_{tot} and R_{part}, are the EV related ones and are not as interesting as the other two, probably because their effect is unambiguous (require starker action when the EV is plugged) and does not interact much with the development of the iterations.

For the sake of completeness, in Figure 12 and Figure 13 we show the swarm plots of the run wall clock time of the episodes; the total time and the time per-iteration respectively. The data are once again divided by C and reward parameter and the individual datapoints are color-coded differently.

First of all, we notice that the episode are position rather similarly in the two graphs, meaning that episodes that take longer in absolute also take longer per iteration. This is natural for those episodes that reach the fixed threshold of 15 iterations, but it is also confirmed for the early-stopping cases. In other words, the cases that reach $\mathrm{mean}(|\Delta V^x|) < 0.01$ before 15 iterations have fast iterations, too, hinting at a parameter setup that synergically causes a "simpler" solution. The only outlier is a case with low C and high penalties, which presents very low wall clock time due to early convergence but a longer iteration, due to anomalous numerical difficulty in the state distribution inference calculations. A possible explanation is that the MDP initialization strategy under stark penalty profiles was optimal by a small margin with respect to relatively distant strategy in terms of action choices, and therefore the introduction of even a small positive C coefficient alters the final profile significantly, leading to a difficult but nevertheless resolutive first iteration. This is an interesting aspect for further research aimed at the convergence properties.

We can notice the most informative factor is C in both the absolute and per-iteration case. Interestingly, for both $C = 1.0$ and $C = 6.6$ two clusters are formed, with sensibly different wall clock time. Interestingly, for $C = 1.0$ all the "fast" episodes are the ones with high R_{therm}, whereas for $C = 6.6$ they are the ones with low R_{therm} – and those are among the fastest to reach a conclusion. They are, in fact, the same four high-C cases that reach $\mathrm{mean}(|\Delta V^x|) < 0.01$ convergence early.

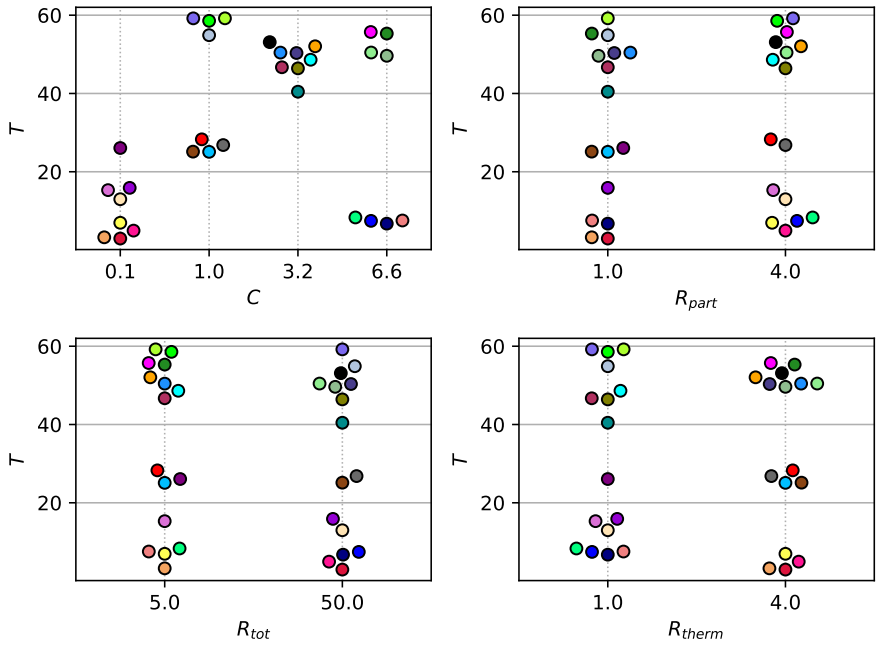


Figure 12: Wall clock time until convergence or τ reached.

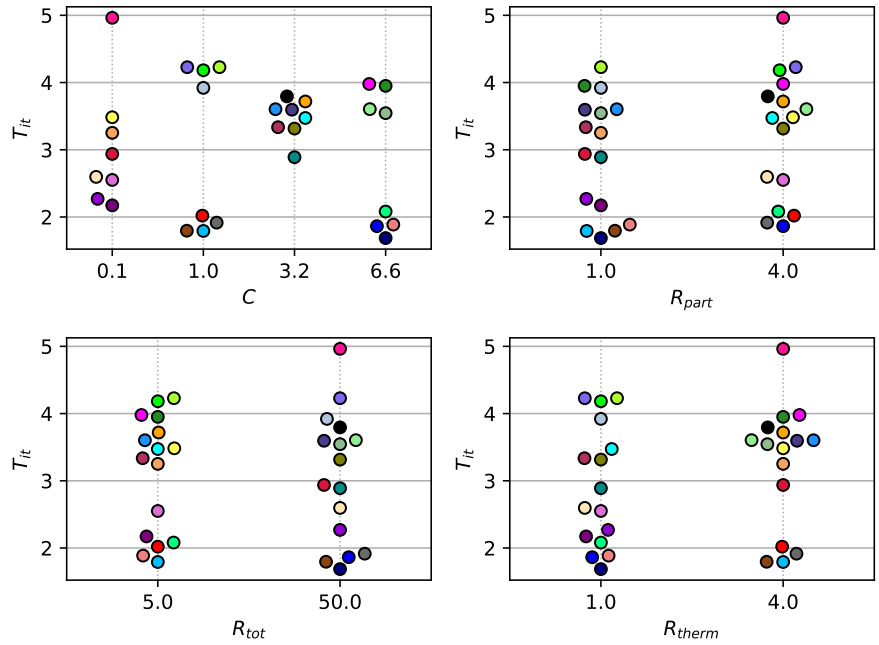


Figure 13: Wall clock time per iteration until convergence or τ reached.

7.2 Strategy visualization

As explained in the previous chapters, a strategy is a map that associates a probability distribution over actions $a_1 - a_4$ to each of the possible states. It is therefore a piece of high-dimensional data and it requires appropriate, domain-specific visualization techniques to communicate the "gist" of the strategy, that is high-level descriptors that communicate very general properties that the strategy often/approximately follows, and that tend to differentiate strategies computed for different scenarios. In this section we illustrate the design of a suitable visualization technique that can communicate summary information about the action taken by the agent and how they are influenced by the state.

We start by analysing the action space. The 4 actions available are the composition of 2 binary actions, "heat" and "charge batteries", as explained above. Therefore, 2 distinct signals are necessary and sufficient to illustrate an averaged probability of taking that actions along some dimension of the state. The first choice that we make is to to visualize the two signals in two completely distinct charts, in this way the two "decisions" are split and can be analyzed separately.

The next design choice is driven by the fact that *time* dimension in the state is arguably the most relevant to unravel in a visualization. In fact, the behavior of the agents is highly time dependent due to the time-variance of the incentives and conditions that vary around them. The presence of the user and its electrical vehicle and the PV production depend on probability curves that vary along the 24 hours. Furthermore, the unitary price curve for electricity is an input variable itself, and to gauge its effect a visualization unraveled along the time dimension is essential. Since the time span used is a day, the solution found is periodic and a polar visualization can be employed to better visualize the periodicity.

Once time is unraveled, we still have five dimensions to analyze. It is impossible to have an explicit, full visualization of the influence of each while mantaining legibility, so we choose to marginalize over all of them and essentially present a hourly average. We choose to show the marginalized probability of taking an action as the color intensity on a greyscale. We can now begin to look at Figure 14 and observe that the two polar plots are divided in 24 radial sectors, representing the hours, and each sector is filled with a plain grey hue. The hue represent the probability of taking the represented action, according to the colorbar shown below. The choice of showing the main output as the background color allows us to add more information about the state, giving some intuition about how the different variables influence the decision.

We can interpret the radius-wise extension of the chart as an additional dimension used to plot an "influence" parameter for each of the additional dimension. What we want is a way to gauge how "relevant" the values of a certain state variable is if trying to explain what the strategy is going to do. This kind of task is very closely related to interpretability for black-box machine learning models [40]. Querying a strategy, in fact, can be viewed as interfacing with an "oracle" that perfectly predicts the action probabilities of the strategy itself using to the state variables as predictors. Here and in the following chapters we will borrow techniques from the field of Interpretable ML to try and "explain" how the strategy is dependent on the state it is responding to, much like one would try to explain how a target variable is dependent on predictors.

Being the direct output of a model-based RL technique, the "raw" strategy itself is a sort of a discrete, exhaustive look-up-table. We incidentally notice that a look-up table is equivalent to a K-nearest Neighbours Regression [41] with $K = 1$, and in fact, in order to practically interface the strategy with the ML interpretability software packages, we wrap it with a K-nearest Neighbours Regressor from the *scikit-learn* Python software package [**?**].

The metric we choose to visualize is the *Permutation Feature Importance* of each of the state variables. This model-agnostic metric offers the ability to confront the various features according to how much

they are relevant to the "prediction of the target variable", which in our case translates to the correct distribution of decisions. This is done by randomly shuffling the values of one of the state variables and measuring how some prediction performance metric, in our case the coefficient of determination R^2, degrades with respect to the base case with no shuffling. The motivation for doing so is straightforward. If shuffling the values has little impact on the prediction performance, it means that the feature has low predictive power. If, on the other hand, shuffling the values completely overturns the predictions, it means that the feature is decisive for a correct decision. The shuffling is done several times to attain a better measurement.

The feature importance of feature j, which we call i_j, calculated with K random repetitions is then quantified as:

$$i_k = s_0 - \frac{1}{K} \sum_{k \in 1 \dots K} s_{k,j} \tag{35}$$

Where s_0 indicates the coefficient of determination of the model with no shuffling, and $s_{k,j}$ indicates the value assumed by coefficient of determination calculated on the k-th repetition of shuffling of the feature j. The minimum possible value for i_k is 0, if the shuffling has no effect on the predictive power, and it has unbounded maximum, as R^2 is unbounded below. These observations are valid only for the particular prediction model and should not be interpreted as a quantification of intrinsic prediction power of the variables; nevertheless, since in our case we are in fact using an exhaustive tabular strategy, we feel that the technique is particularly poignant.

The evaluation of feature importance is done separately on strategy data for each separate hour, and the obtained mean values are plotted around the polar chart as lines, with higher distance from the center representing higher values on a marked grid. When the probability of taking an action happens to be approximately 0 or approximately 1 for a certain hour across all other state variables, the point is not drawn because features other than the time are meaningless. Figure 14 shows an example of a final figure obtained with the procedure described here.

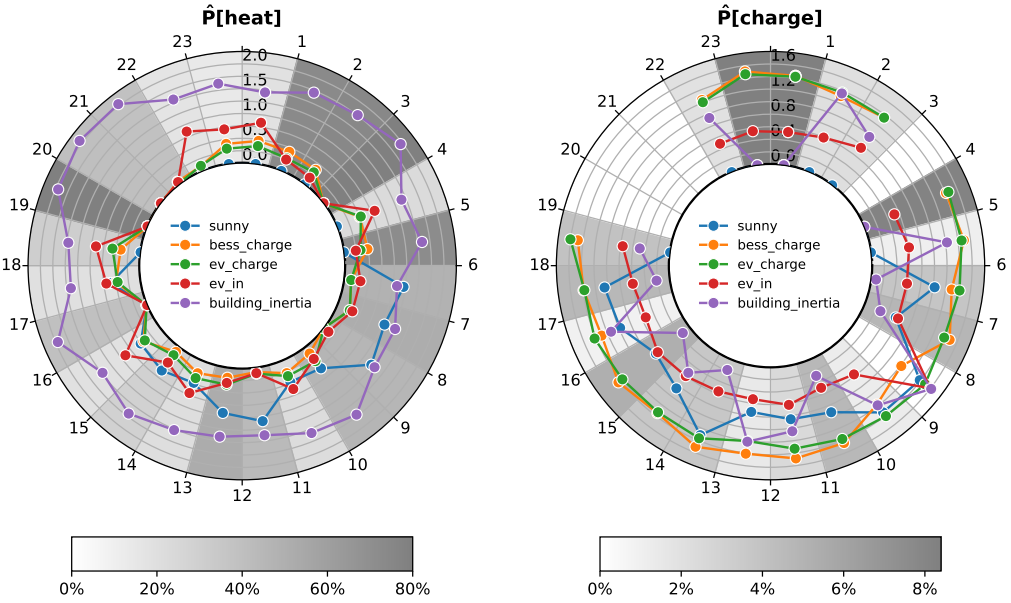


Figure 14: Visualization of a strategy.

8 Parametric analysis

Having in place a tool able to compute sensible behavior for an agent, one of the most interesting aspects to investigate is the sensitivity of this behavior with respect to the various parametric quantities that influence the rewards it is seeking to maximize. The reward mechanism was already introduced in 3.5. Similarly to what was done for the convergence analysis, the quantities that will be varied are the following:

- The C coefficient (see Eqq. 25 and 29)
- R_{tot}
- R_{part}
- R_{therm} (see Sec. 3.5)

The sensitivity analysis will involve varying each of these parameters in a set of values, computing the strategy for each combination on the grid using the technique shown in 5.

When trying to visualize the results, we face problems very similar to the ones explained in 7.2: the strategies computed are highly multidimensional data items that potentially describe complicated inter-dependencies. The situation here is even more complicated, as we are adding more dimensions – the penalty constants. Since the very aim of the analysis is showing the influence of the penalty constants, the corresponding axes must be the ones unraveled in any representation. We can only hope to show one scalar quantity describing a whole strategy at a time. Arguably, the two most descriptive global scalar properties that can be computed for a strategy are the mean frequency of the $heat$ and $charge$ actions, marginalized over all the states. Clearly, by doing so, more complicated dependencies of the

action on the particular state are lost, but nevertheless the mean proved to be an adequate measure to express the general tendencies of the strategies. In light of these consideration, we choose as the main means of visualization a Partial Dependence Plot of these global averages over the varied quantities.

Realistically, the maximum number of unraveled axes for a single chart is 2, so the whole representation will be composed by several 2D contour plots in which we show two varied quantities explicitly and marginalize over the others. In order to choose which axes to show, we notice that the first item, the C coefficient, is different in nature from the others since it scales a coordination penalty and not an individual reward. It is also perhaps the most interesting, as it is the quantity that couples together the agents giving rise to a stochastic game, as explained extensively in 4. Our choice for this analysis, therefore, is to show 3 2-D Partial dependence plots always showing C in the x-axis and the other quantities in the y-axis one at a time.

In the following subsections we will present the outputs of the analyses for a few profiles of weather and user consumption.

8.1 No elicitive absorption case

The standard scenario we will start our analysis from is computed with the following characteristics

- Radiance clusters transitions fitted on the year 2016 @ $45°00'00.0''N\,8°00'00.0''E$
- No absorption other than the one caused by the agent
- $\gamma = 0.9$
- $\tau = 5$

The second hypothesis is a strong simplification that we use in this first standard scenario to offer an easily interpretable baseline that will hopefully make it easier to interpret more complicated phenomena in the following scenarios. The sensitivity analysis, as presented above, is shown in Figure 15.

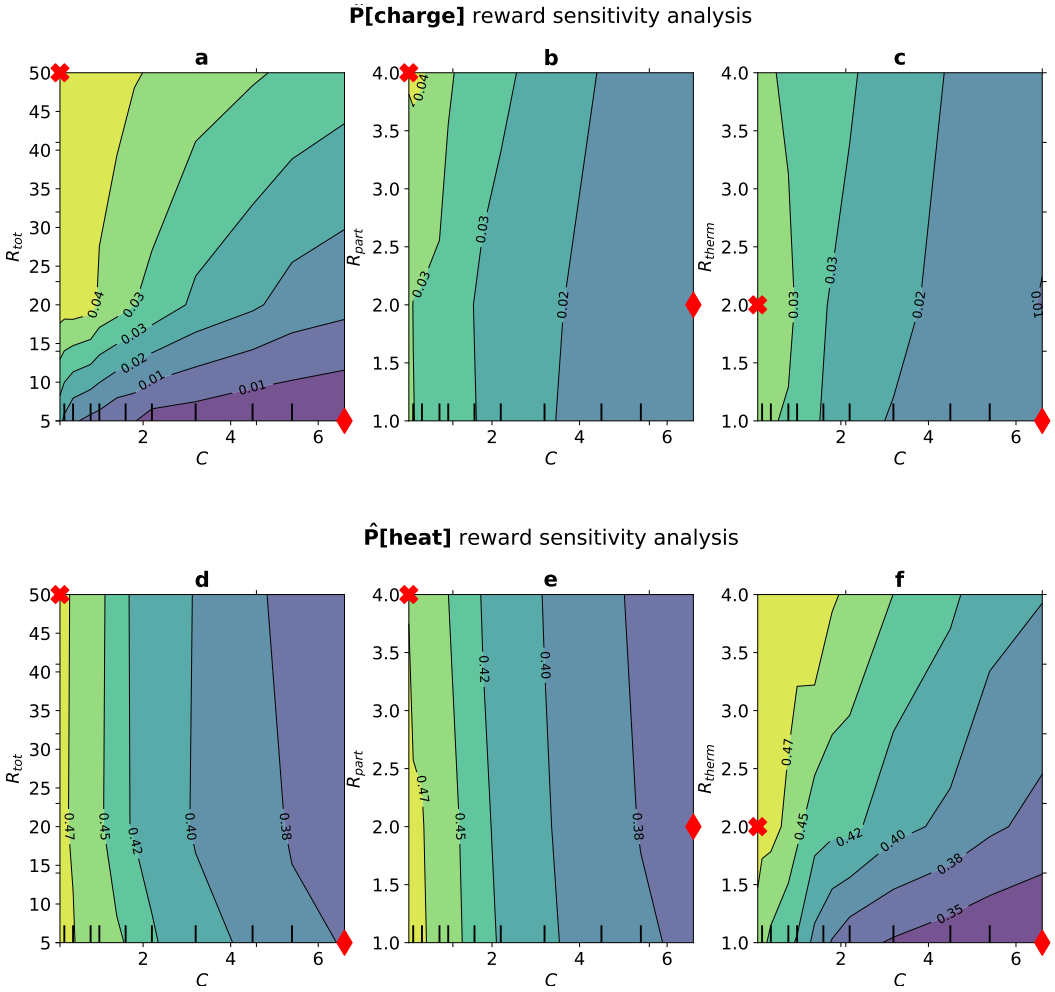


Figure 15: Partial dependence display for scenario with no elective absorption and standard weather.

The sensitivity of the mean probability of charging is shown in Figure 15$a-c$. Figure 15a shows the dependence on C and R_{tot}. The relationship is very clear: the strategy becomes more "aggressive", that is, more inclined to use the charging action in general, with increasing R_{tot}. The effect is mitigated by an increase in the C coefficient. The same dependence, but in a starkly less pronounced magnitude, is observed with R_{part} in Figure 15b. R_{therm}, on the other hand, does not show particular influence on the charging action, as one could expect. It remains clear from Figure 15c that the C coefficient exerts the designed disincentive action.

The following row of figures shows the same quantity for the thermal action. Conversely from the previous row, R_{therm} exerts the most influence on the probability, causing a dependence similar to the one that R_{tot} had on the charging action. The charging-related rewards do not seem to have much influence, but it is possible to discern a slight tendency to reduce the thermal action probability as they increase. This can be explained in the following way: as the charging penalties are increased, the charging action is used more, and the thermal action, that can be more easily shifted, is more "dispersed" along time as a consequence, to avoid charging and filling the thermal buffer at the same time.

In Figure 15 we pinpointed two particular conditions, which we analyze more closely with the illustration method presented in 7.2:

- Marked by $\times$: High penalties, low C. Displayed in Figure 16.
- Marked by ⧫: Low penalties, high C. Displayed in Figure 17.

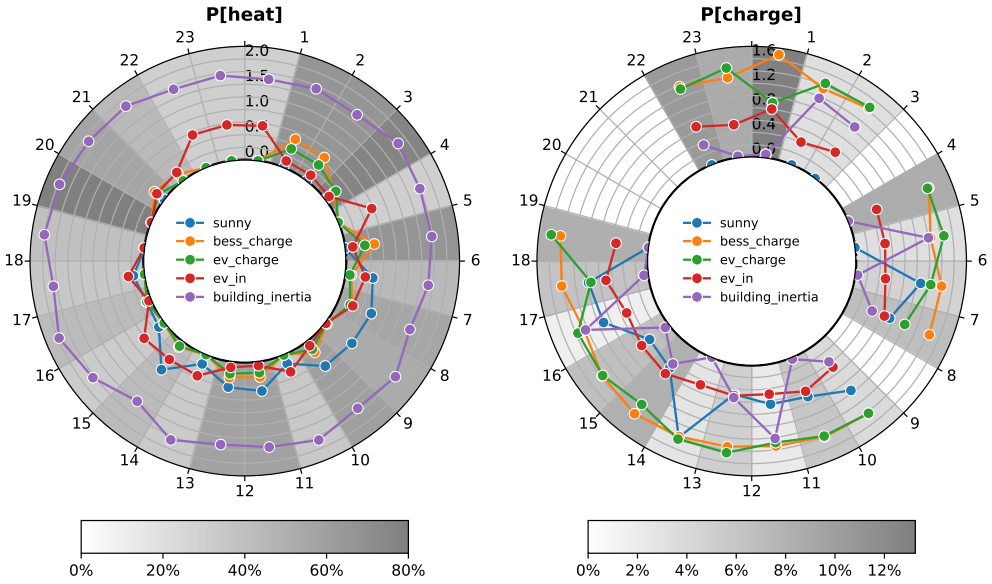


Figure 16: Analysis of strategy marked by $\times$. High penalties, low C.

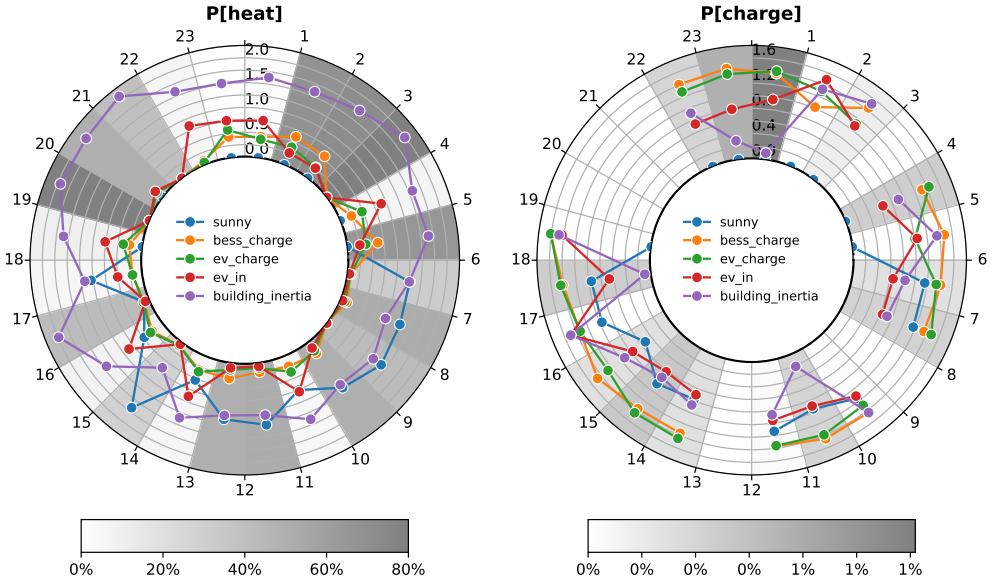


Figure 17: Analysis of strategy marked by ⧫. Low penalties, high C.

The effect brought by the parameter change is clear.

- Thermal: the general probability of taking the thermal action is lower, and in particular lowers further when already low. The feature importance also changes: with high penalties and low C, the decision is essentially driven by the state of the thermal inertia only, with the presence of the EV playing a small role in the evening hours (remember that when the EV is out, the thermal penalty is not experienced, as the user is modeled as being out of home). In contrast, the decision appears more faceted with low penalties and high C, with the decision being influenced by PV production. The most straightforward interpretation is that the agent speculates on the BESS being charged the following hour, so that it will be able to action the heat pump without grid impact.
- Charging: The general probability of charging is drastically reduced when going from $\times$ to $\blacklozenge$ (notice the scale in the colorbar). The parameter describing the presence of the EV increases in importance, which is a symptom of the agent choosing to the charging action just in time to avoid the R_{tot} penalty.

8.2 Elicitive absorption case

We now repeat the same analysis with the same parameters, with the only difference being the use of the curve of elicitive consumption on the agents' part, shown in Figure 5. The results of the analysis are shown in Figure 18.

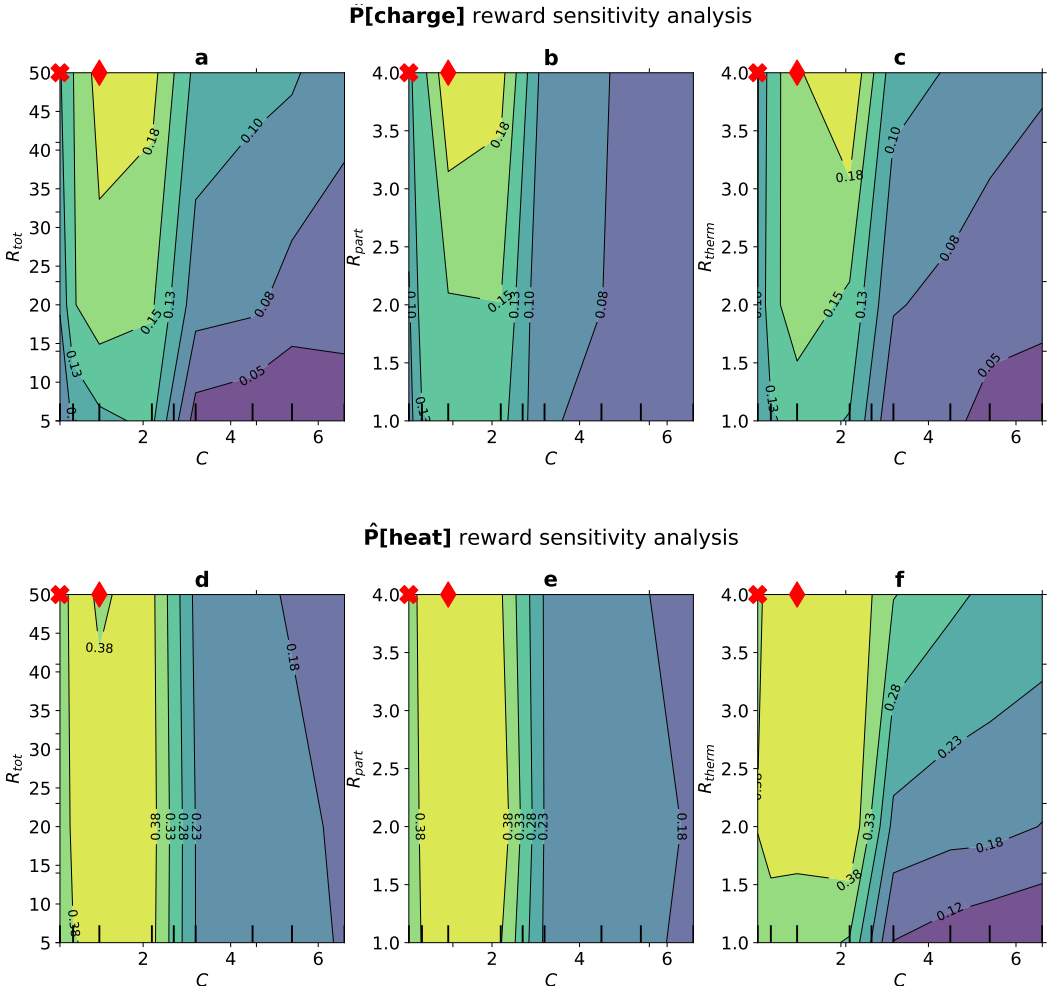


Figure 18: Partial dependence display for scenario with elicitive absorption and standard weather.

The results here are more interesting, particularly on the charging action, as the C coefficient seems to have a non-monotone effect: an initial increase in C, causes a stronger use of the charging action on average; the effect is then reversed and the usage of the action reduces as C reaches values $\gtrapprox 3$.

In order to try and explain this effect, it is useful to analyze two strategies, again marked by $\times$ and ⧫.

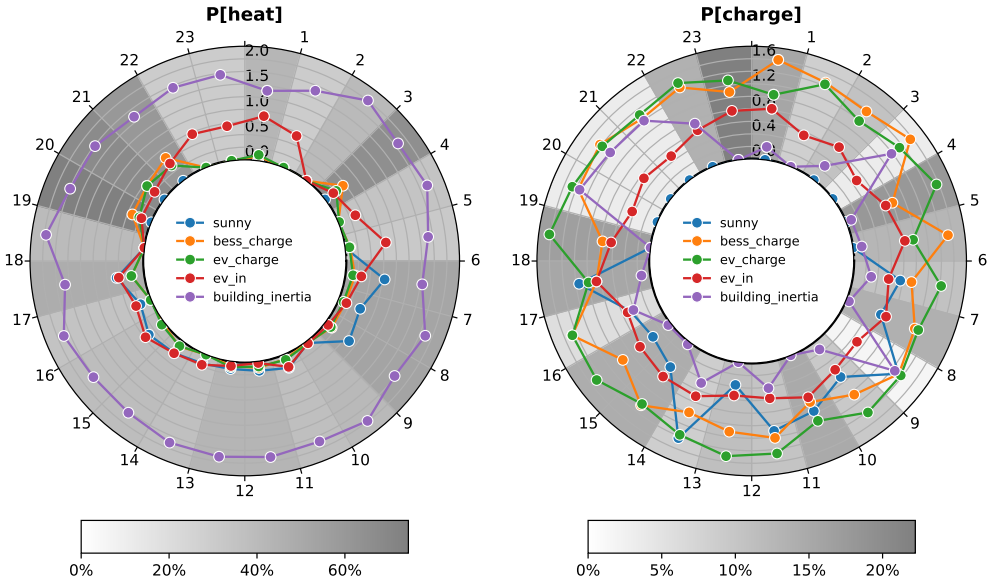


Figure 19: Analysis of strategy marked by $\times$. High penalties, low C.

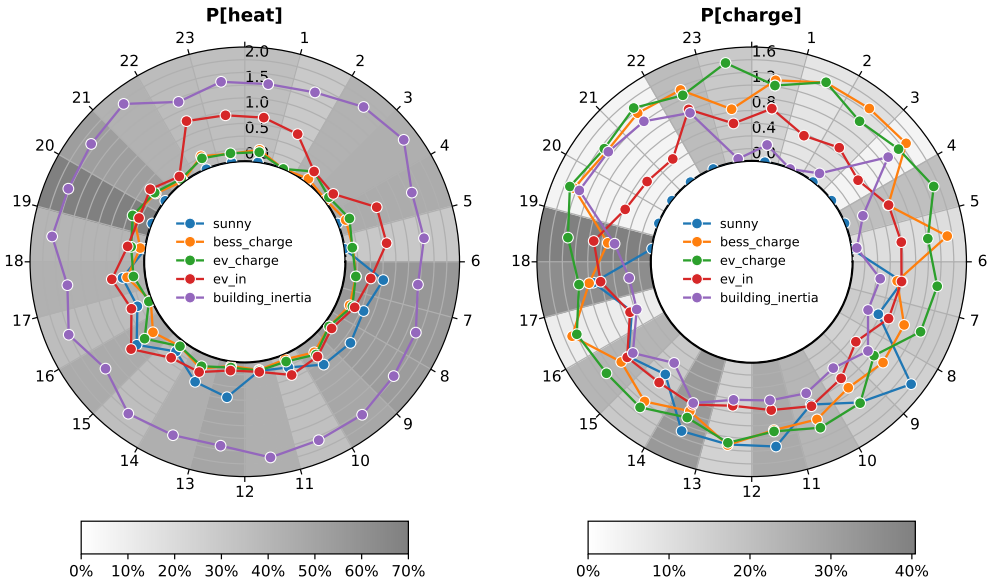


Figure 20: Analysis of strategy marked by ⧫. High penalties, intermediate C.

Here we see that going from $\times$ to ⧫ the charging action becomes more intense (notice the levels represented by the colorbars), and in particular, the action is intensely selected at hours 17-18, which is just before a big peak in deliberate consumption. The interpretation is straightforward: the agents uses a moment of low consumption to charge the battery and shave the consumption peak exposed to the grid in the following hours.

9 Flexibility in response to incentives

In this section we will expose, as an example of the usage of the methods of this work as a flexibility estimation tool in response to incentives, an analysis for a price curve composed of a base price of 1 and a discounted period going from hour 12 to hour 18. This analysis provides a first view on the utility of the tool as an estimation and management tool for Community Managers. By modifying the price curve, managers can estimate the ability to steer the consumption of a flexibility pool. This approach, has to be complemented by an opportune fitting of the parametrization of the model on the particular community steered. Possible ways of obtaining parametrizations could involve experiments on a pilot project or surveys about user preferences. This remains an interesting, but open question outside the scope of this work.

Naturally, the space of the possible hourly price curves is very big. Here we will have to limit ourselves to an example scenario of incentivizing residential consumption in the afternoon period with a discount to showcase and analyze the capability of the tool with a significand case study, but the tool is naturally capable of evaluating different price curves. Strategies were found for different levels of the following parameters:

- The C coefficient
- The penalty levels (according to Table 4)
- The price in the discounted period.

Table 4: Penalty scenarios for flexibility analysis.

	R_{tot}	R_{part}	R_{therm}
Low	10.0	1.0	1.0
Medium	20.0	2.0	2.0
High	50.0	4.0	4.0

According to the first two parameter, a "baseline" consumption can be identified, that is, the consumption if the price is kept fixed at 1 for the whole duration of the day. The raw data of baseline consumption are summarized in Figure 21. Recall that these data are directly extracted from the calculated strategy for the aforementioned conditions according to Equation 22, with the example user consumption data of Figure 5.

The figure also presents a grey shaded area for visualizing the period that is subject to incentivization. To aid in a clearer exposition of the effect, the day is subdivided in Morning/Middle-day/Evening according to the subdivision induced by the incentivized period. This reduces spurious/random effects hour-by-hour when considering the change in energy consumption. The scenarios with modified prices are calculated. In the following figures, we will show the percentage change in consumption in each of the time slots. The first two figures, Figg. 22 and 23, show the partial dependence of the consumption change respectively on C and on the penalty levels. These plots, naturally, are marginalized over all the prices.

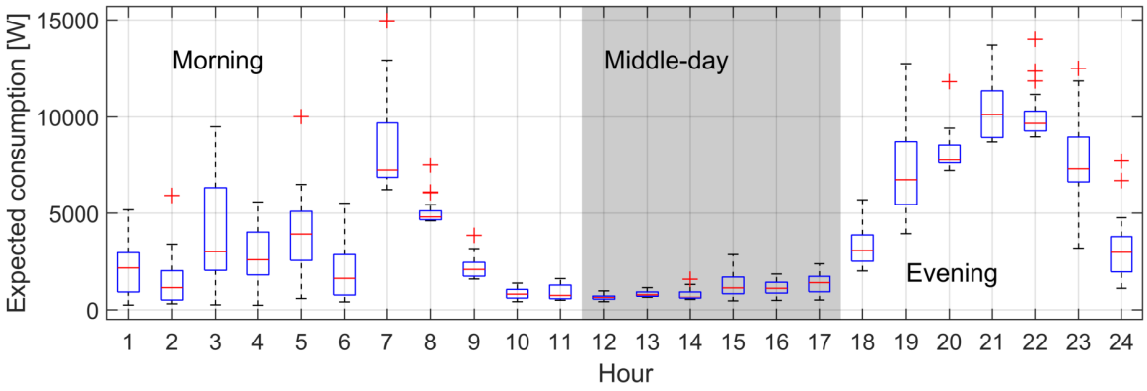


Figure 21: Baseline consumption, for all the computed cases.

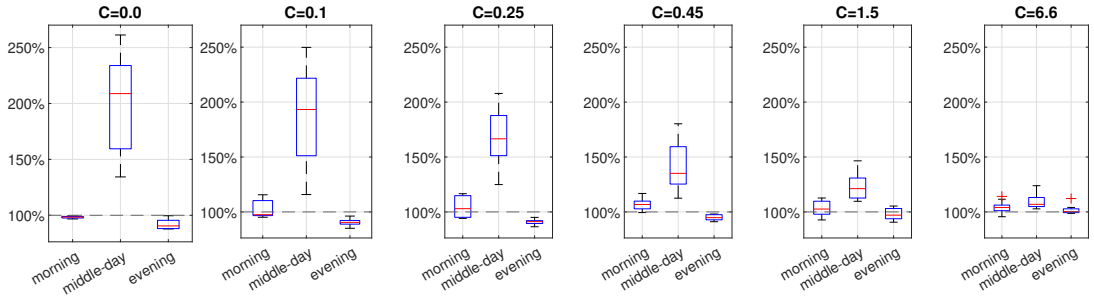


Figure 22: Change in consumption box plot for different Cs.

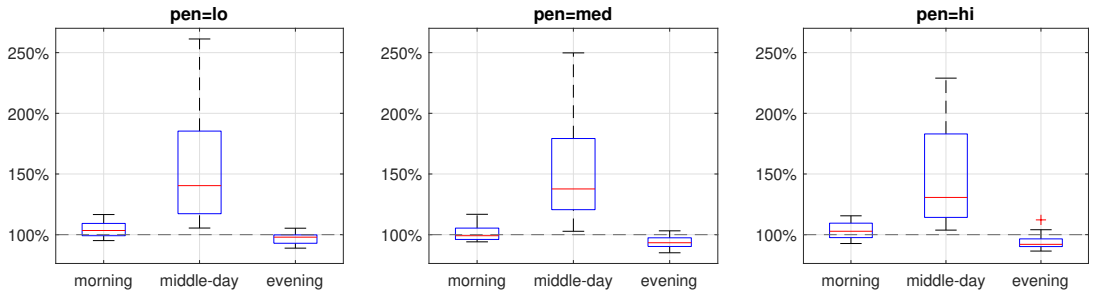


Figure 23: Change in consumption box plot for different penalty levels.

The C coefficient, has a marked effect on the change in consumption. Higher Cs, and therefore high disincentive to coordination, progressively limits the effect of the price changes. The effect of the penalty levels is less marked, but is nevertheless present. Lower penalties (and therefore higher flexibility) translate in bigger relative increases in consumption during the middle of the day.

We can now turn to the effect of changing the price level in the incentivized period. As expected, the effect is quite marked. The results are summarized in Figure 24.

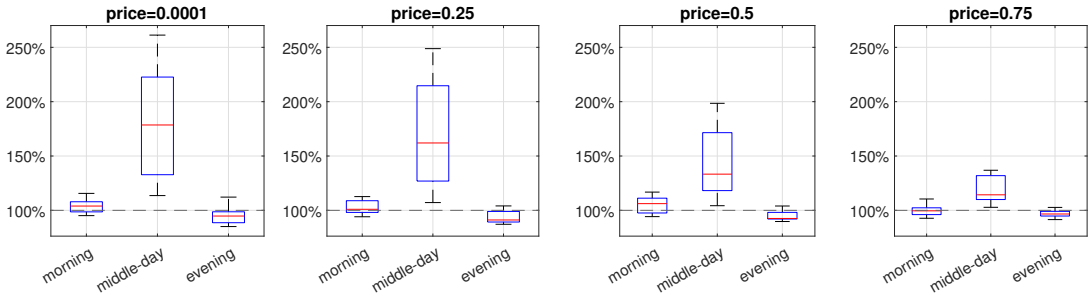


Figure 24: Change in consumption box plot for different prices.

In order to offer a summary visualization of the effectiveness of the incentive, we need to define some scalar merit parameter describing the entity of the consumption shift along the entire day. Recalling Equation 22, and calling S_H the subset of S where the hour of the day is H, we write the expected hourly consumption given policy π as:

$$\hat{\mathcal{E}}_H(\pi) = \sum_{s \in S_H} \sum_{a \in A} \hat{\mathcal{E}}(s,a) \pi_{s,a} p^{\infty}(\pi, s) \tag{36}$$

We now call π_p the policy calculated for incentivized price p. The policy is calculated given a parametrization in terms of coefficient C, flexibility parameters, etc., but we do not indicate this in the notation in order not to further weigh it. In the following equations, when two policies π_{p_1} and π_{p_2} are mentioned, they are always calculated with identical parametrization and the only difference is the price curve, and in particular the relative price levels in the incentivized period as defined previously in this chapter. We define a *Total (expected) displaced consumption* as:

$$\Delta^p_{\hat{\mathcal{E}}} = \sum_{H \in 1:24} |\hat{\mathcal{E}}_H(\pi_p) - \hat{\mathcal{E}}_H(\pi_1)| \tag{37}$$

We can now show the total incentive paid, the total energy displaced and the incentive per unit displaced consumption for each of the C-penalties parametrization, according to the incentivized price in the following Figures 25, 26 and 27.

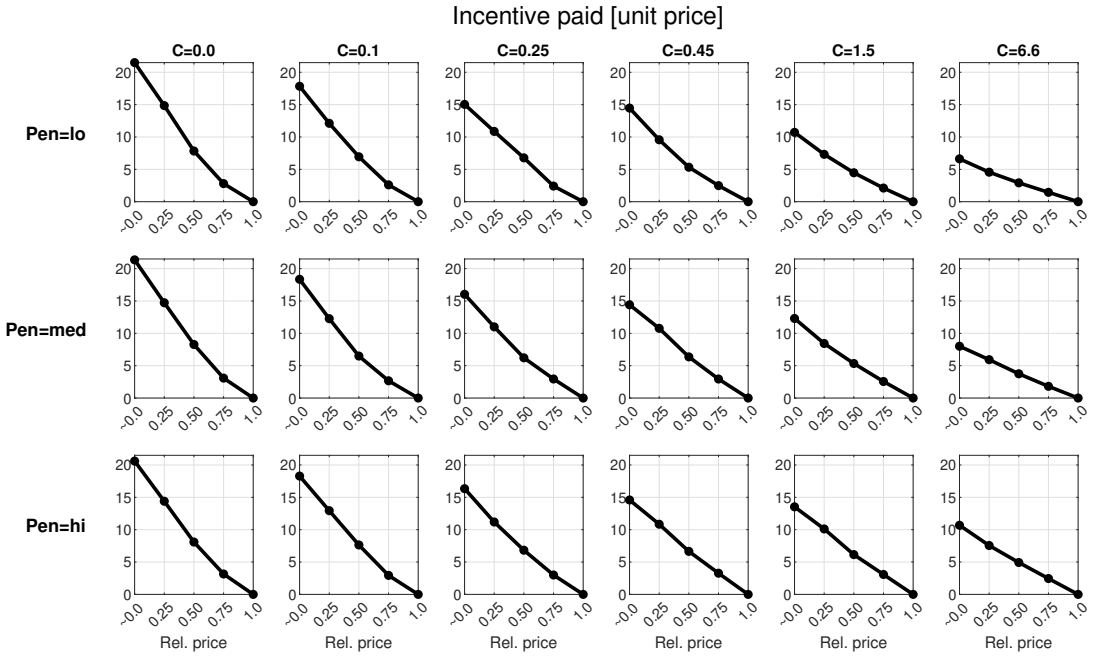


Figure 25: Incentive paid.

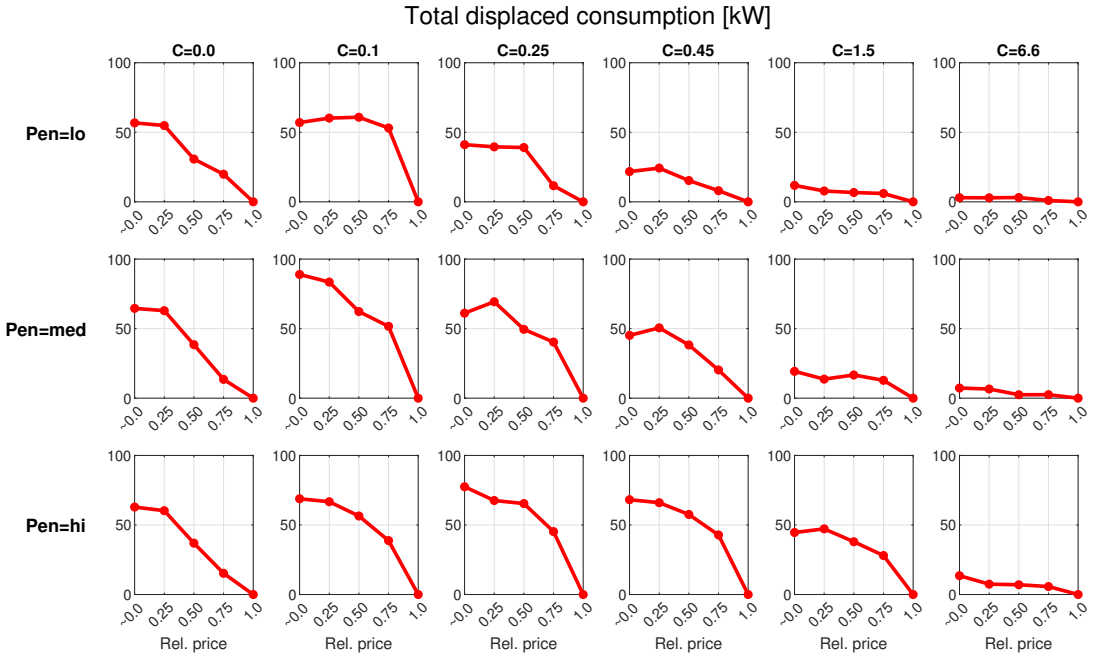


Figure 26: Displaced consumption.

Recall that the economic quantities are expressed in an adimensionalized unit price, corresponding to

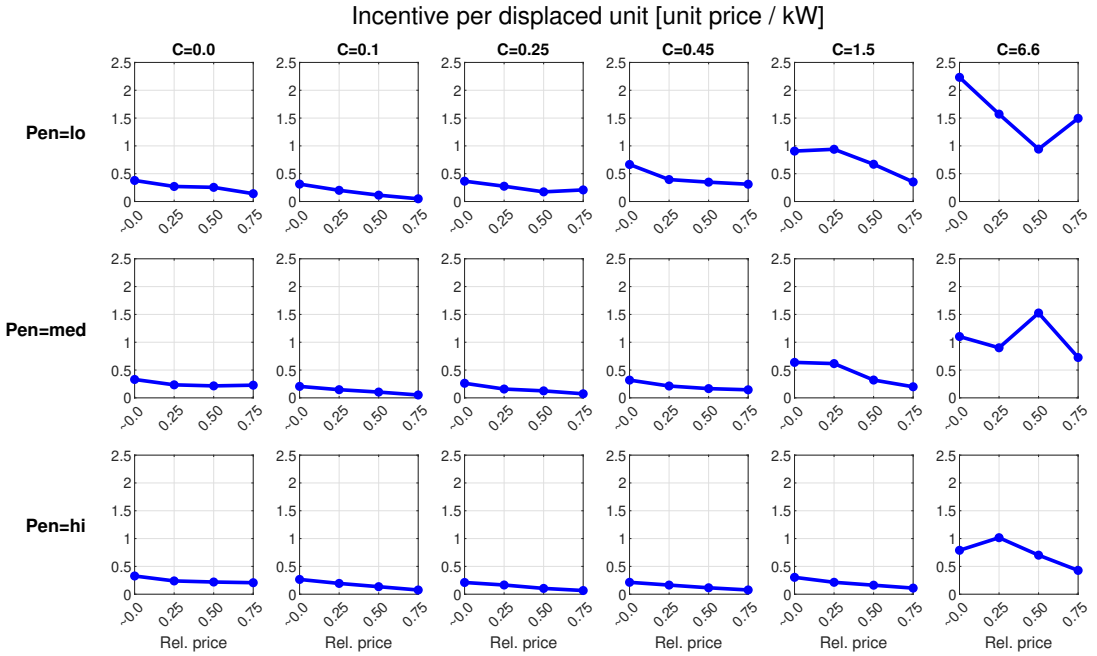


Figure 27: Incentive per unit consumption displaced.

the "base price" of one kWh agreed with the counterparty. These example calculation pose the basis for the closure of the economic problem of the community manager: according to how exactly the obtained curve is remunerated by the counterparty, the community manager can evaluate the cost of consumption shifting he experiences. A few aspects are, in the author's opinion, of particular interest:

- the total displaced consumption tends to plateau with the reduction of the price;
- the marginal cost of displacement tends to increase slightly with the reduction of the price;
- the increase of the C coefficient has initially a gradual effect on the incentive per displaced unit, but, as expected, very high values of C make the shift very stiff as the agents already tend to distribute consumption as much as possible.

10 Grid impact simulation

10.1 Software structure

In this section, we will proceed to simulate a power flow in a test grid, where end-nodes are populated with automated agents. This is a further demonstration of the potential utility of the tool for smart grid managers, in particular with respect to grid constraints. Our formulation, including the C coefficient to modulate the collective influence of the agents on the network, allows the manager to estimate the voltage distribution when agents are acting in the grid and change thereof for different parametrization. It is conceivable, and a possibility for future work, to employ higher C coefficient to offer regulation services for the DSO, in exchange for a compensation (generally speaking, an increase in C further from the level that is found to guarantee grid stability for the particular situation pushes the solutions away from pure, individual benefit, which must be compensated).

In order to evaluate the impact of a given policy on the grid, a simulation framework incorporating a power flow solver was developed. In this section we will illustrate in detail how a simulation is performed, starting from a policy.

A policy π is a map from a measured state to a distribution of action probabilities. Modeled agents use it as their method to decide which action to perform. Aside from the agent model, which determines how the state evolution happens as described in Chapter 3, the information to supply to perform a simulation of a policy π is:

- A list of buses where smart agents are installed (b)
- An electrical description of the grid, suitable for the utilization with the Pandapower package
- A list of starting states for the agents (S_0)

A scheme of the simulation framework is shown in Figure 28.

The simulation is performed for a configurable number of steps, n. At each step, the agents measure their own state, sample the appropriate action distribution from policy π and perform the vector of actions, $\bar{a}$. According to the simulated outcomes of the random events that determine their realized state transition, the agents obtain their updated state and the energy units exchanged with the grid. The vector of these energy exchanges, $\bar{E}$, is used in the power flow solution to obtain the state of the network. As discussed in 3.2.2, all the agents are subject to the same evolution of the "natural", shared part of their state. The implementation takes this into account, and the transition of the shared part of the state is forced to be the same for all the agents, that then evolve the private part of their state independently of one another.

At each simulated timestep, all meaningful quantities (actions performed, state history, energy exchange and the solution of the power flow) are saved to be used in further analysis a posteriori.

10.1.1 Simulation output

The output of the simulation is delivered in tabular form. The recorded quantities are the following, with one record per each timestep, as shown in Table 5.

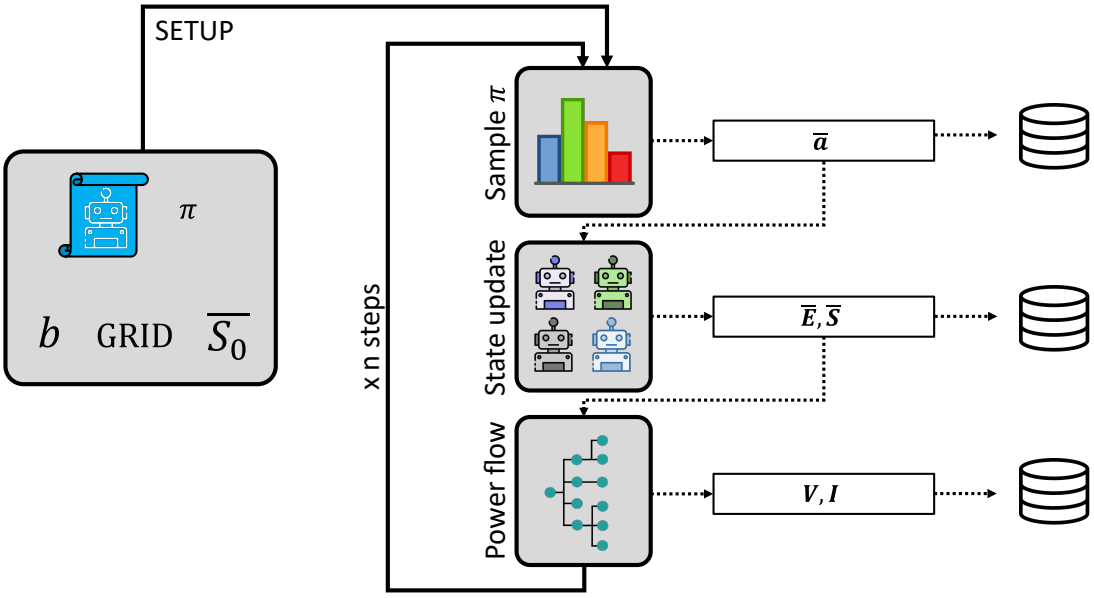


Figure 28: Scheme of the cosimulation framework.

10.2 Network and loads

To deploy our strategy in simulation, we will use a test synthetic low voltage network according to [42], created with the PandaPower package. The network schematic is shown in Figure 29. The data of the lines and the transformer of the network are found in Appendix A.

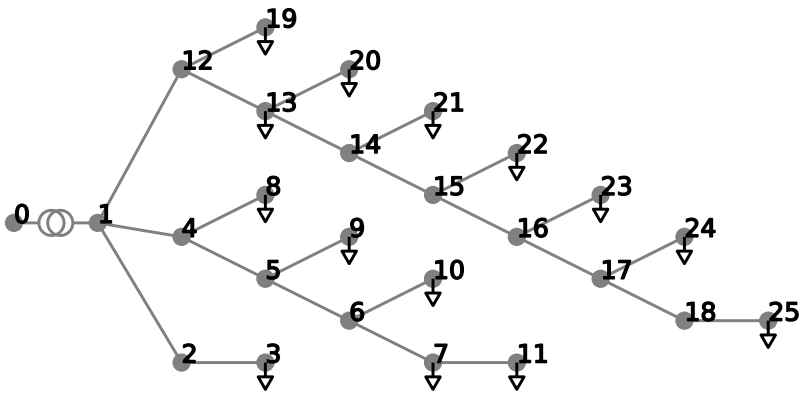


Figure 29: Test network.

The only loads modeled in the grid will be the smart agents executing the strategies. One agent is inserted per loaded bus (indicated by a triangle in Figure 29) The constant power absorbed by the agent in a single timestep increases by $10kW$ for each "energy unit" absorbed from the grid according to the model, which means that the individual energy unit is $10kWh$ since the timestep is $1h$.

Table 5: Recorded simulation output (one record per timestep).

	Quantity	Unit	Property of
Electrical	Voltage magnitude	P.U.	Bus
	Loading percent	%	Line
State	Time	h	Global
	Sunny	bool	Global
	Bess Charge	int	Agent
	EV Charge	int	Agent
	EV in	int	Agent
	Building inertia	int	Agent
Action	Action	cat	Agent
	Energy absorbed	int	Agent

10.3 Analysis

The most interesting metric to analyze to gauge the effect of the strategies on the network is the voltage. In this section, we will compare how different parametrizations impact on the voltage profile at the buses.

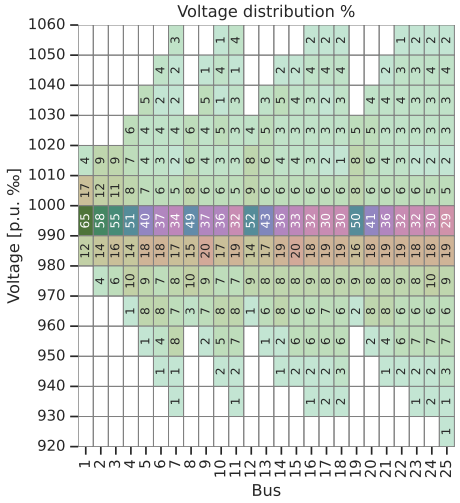
The simulations can be easily and quickly run for all the computed strategies. To illustrate the effect of the C parameter, we select the following fixed conditions:

- $R_{tot} = 20$
- $R_{part} = 2$
- $R_{therm} = 1$

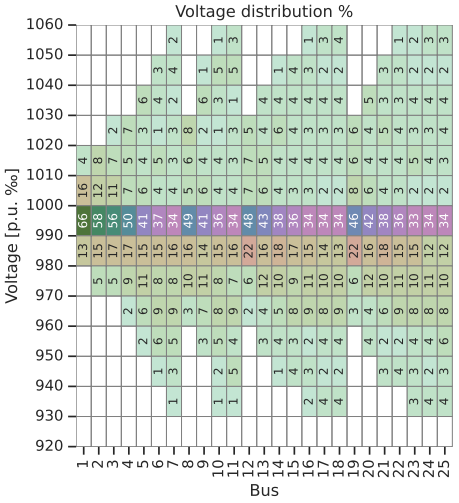
10.3.1 Effect of the C coefficient

In Figure 30, we show the empirical distribution (histogram) of the voltage experienced at the buses, for the simulation of strategies obtained in the standard scenario. We compare the simulations run for different levels of C.

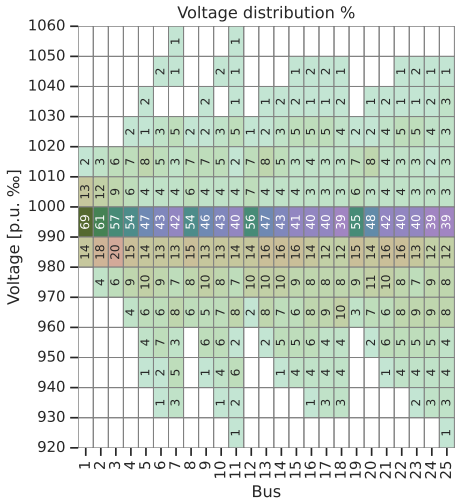
Most of the time, the voltage is concentrated in the 0.990-1.000 p.u. range, with values around 30% on the most peripheral buses. Increasing C has the the effect of concentrating the distribution even further: it is particularly noticeable that the 0.980-0.990 p.u. range gets progressively narrower.



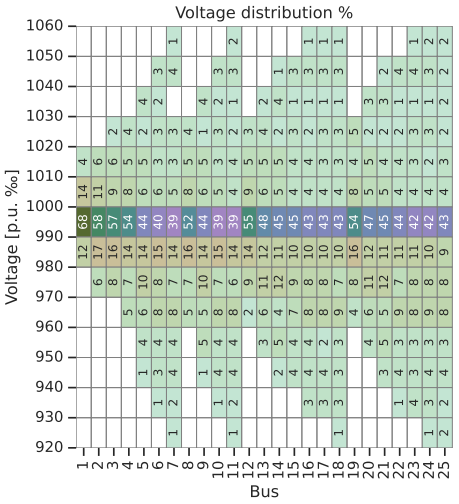
(a) Test case bus voltage histograms, $C = 0.1$



(b) Test case bus voltage histograms, $C = 1.0$



(c) Test case bus voltage histograms, $C = 3.2$



(d) Test case bus voltage histograms, $C = 6.6$

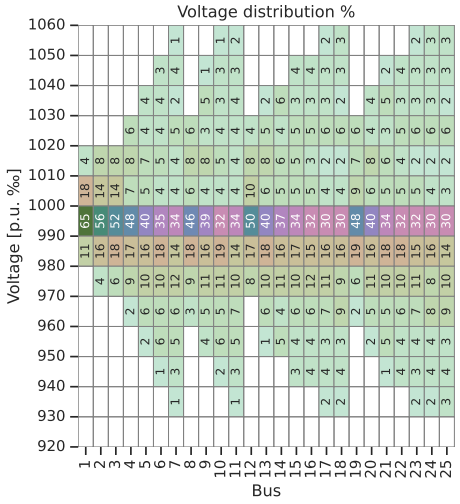
Figure 30: Analysis of Voltage vs C (part 2).

This showcases how the C coefficient, as mentioned above, can be used to stabilize the voltage levels around the nominal values, offering a better distribution with less voltage surges and drops. The C coefficient, in fact, modulates the collective term in the objective function (Equation 29), as underlined previously, incentivizing an improved voltage distribution.

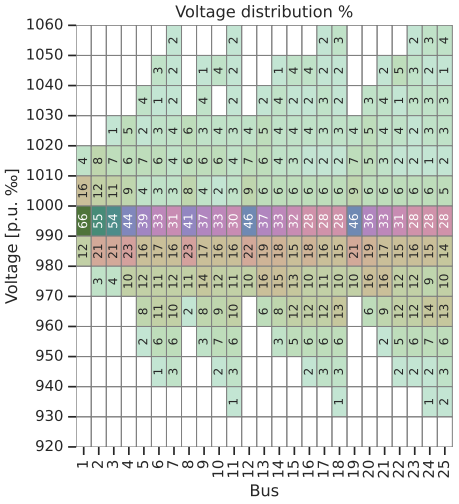
10.3.2 Effect of flexibility

We now evaluate the effect of flexibility on the voltage. We keep fixed $C = 0.1$ and evaluate three reward profiles increasingly punishing for discomfort:

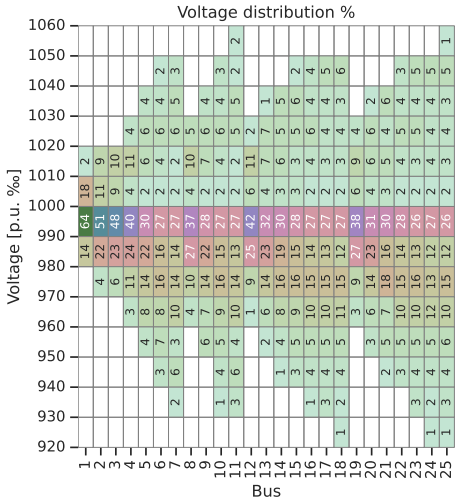
- $R_{tot} = 5$, $R_{part} = 1$, $R_{therm} = 1$ (high flexibility)
- $R_{tot} = 20$, $R_{part} = 2$, $R_{therm} = 2$ (medium flexibility)
- $R_{tot} = 50$, $R_{part} = 4$, $R_{therm} = 4$ (low flexibility)



(a) Test case bus voltage histograms, high flexibility



(b) Test case bus voltage histograms, medium flexibility



(c) Test case bus voltage histograms, low flexibility

Figure 31: Analysis of Voltage vs. flexibility.

The results are shown in Figure 31. The effect is very clear: as the users' preferences are stiffer towards discomfort (which translates to high punishment values), the voltage distribution broadens and shifts

towards lower values. This is easily interpretable, since stiffer punishments mean less reward margin for the strategy to reallocate the load away from request peaks. Therefore, in a real implementation, if the users' preferences are too stiff, the power quality might suffer; again, the C coefficient can be used to mitigate the effect as explained above to compute an appropriate strategy avoiding disruption, improving the situation.

Part IV
Conclusion

11 Summary

Training Multiagent Strategies for the Grid is a project aimed to provide tools and insights for tackling the management of electrical flexibility, a challenge made more and more pressing by the need for the integration of renewable energy sources in the electrical grid down to the distribution level, and increasingly actionable thanks to schedulable loads such as heat pumps and EVs and the technological control and monitoring substrate offered by the smart grid.

The approach taken in this work is integrally distributed: each agent manages energy assets on behalf of an owner autonomously, and has to do so optimally in a stochastic environment. This approach has the invaluable practical advantage of not requiring the operation of private loads by a central remote controller out of the jurisdiction of the owner, but presents the challenge of overcoordination among similarly interested agents managing similar assets without explicit coordination. This potentially aggravates the burden on the distribution grid instead of relieving it, leading to undesired grid disruption, that penalizes all the agents. This strategic background can be modeled as a Stochastic Game, a general framework often used in Multiagent Reinforcement Learning. A detailed description of how this is accomplished is offered in this report. A salient characteristic of the modeling presented is the presence of a penalty for overcoordination through a coefficient, C, that allows to scale the importance of this effect on the reward profile of the agents, incentivating more "conservative" strategies. Stochastic Games are in general difficult to solve, as any learning agent has to face an intrinsically deeply nonstationary environment, in which the behavior of the other agent changes in response to the change of his own. After an attempt at implementing a generic technique based on Replicator Dynamics, we turned to a more specific approach by leveraging the peculiar structural characteristics of the problem at hand and we managed to introduce an approximation method inspired by a technique from the literature on card games of imperfect information. The hypothesis on which the introduced technique relies is that the agents could be considered symmetrical (sharing the same state space and reward structure). This is not in contradiction with a modeled reality in which the actors do, in fact, manage similar assets under the same incentives, and therefore the strategic insights remain valuable. The main limiting factor is given by the state space, that suffers from the *curse of dimensionality* that appears as one tries to discretize more finely the various state parameters, such as the SOC of the batteries or the timestep. Nevertheless, the state space tractable by our technique is large enough to adhere satisfactorily to the modeled reality and to be representative of the main strategic features.

A key computational task to perform to apply the method is to derive the distribution of states of the other agents from the perspective of an agent whose state is known under a certain policy, which is necessary to compute appropriately the next iteration of the policy at that state. This step takes a large part of the

wall clock time when running the algorithm and has to be implemented carefully to minimize the time and memory consumption, therefore a section is dedicated to the description of this task.

The last part of the work was dedicated to analyzing the performance of the method and perform various analysis on the results. The convergence of the method iterations are analyzed, underlining the role of the various parameters on the stabilization of the strategy. The strategy profiles obtained are pieces of intrinsically high-dimensional data and considerable thought has to be spent on their presentation. We introduce a dedicated visualization technique for the strategies, showing its mean values through time while simultaneously giving information about the importance of the state parameters on establishing the strategy at a particular timestep. Furthermore, Partial Dependence Plots are used to capture the sensitivity of the global mean of the strategy to changes in the reward scheme, which is important in illustrating the potentiality of the end users' flexibility in accepting discomfort when trying to minimize grid impact. The effect of hourly price incentives on consumption shift is then evaluated for a significant case study. Finally, the agents enacting the computed strategies were simulated in a power flow model on a test distribution network. This allowed us to gauge the effect of the different strategy parametrizations on the voltage profile, confirming the main intuitions qualitatively and providing a valuable quantification tool.

12 Outcome of the project

As underlined throughout this report, the detailed parametric characterization of the model underpinning the proposed tool has a faceted significance. Summarizing, the two main "usage modes" we suggested for the applications are:

- estimation of flexibility potential by using a price signal;
- estimation of grid impact according to the chosen parametrization.

The proposed tool has several advantages compared to approaches found in the literature. In particular, we underline the following:

- it accounts for stochastic nature of renewable energy, does not rely on opinionated behavioral modeling frameworks such as prospect theory [43];
- agents can be prosumers, and not exclusive producers or consumers [44];
- it is not limited to mechanism design or other abstract formulations but models the environment and proposes a strategy [45];
- grid impact is taken into account through the presence of the C factor, which can be used to tune the behavior to avoid grid disruption. This aspect is typically not accounted for.

The main limitations of the approach reside in the agent symmetry hypothesis, necessary for obtaining concrete solutions but at the cost of modeling accuracy, and the necessity to fit the rewards parameters on actual user preferences. The former aspect could be the object of further investigation. In a pilot project, the latter aspect could be appraised through user satisfaction surveys. The essential steps to perform in a pilot project could be the following:

PHASE 0: offline computation of several, localized strategy profiles for various parameters and seasonality.

PHASE 1: wiring and set-up of agents able to actuate the flexibilities (EV, BESS, heat pump) and to sensor and monitor their state. The agents should be able to read the selected strategy profile and sample its actions from it.

PHASE 2: execution of strategies with various levels of penalties, and satisfaction survey to understand how the model fits actual user preferences.

PHASE 3: measurement of grid voltage with various levels of C, to appraise the effect of this collective coefficient, while keeping monitored user satisfaction.

PHASE 4: application of price incentives, measurement of flexibility steering with respect to a reasonable base case. Survey on the balance between user compensation and induced discomfort (if any).

13 Outlook and next steps

This work introduces a complex and capable model, with the aim of generality. Several further research alleys can be explored. The most important, in the author's opinion, are the following:

- Fitting the parametrization of the model on actual test cases and pilots; most importantly, evaluate in reality the range of perceived penalty on the user's part, and therefore their flexibility, informing the rewards to be set in the model.
- Investigating business models and regulations with the aim of offering voltage control services to the DSO, through the manipulation of the C factor as previously explained.
- Investigating the multi-population case; while obviously interesting, this is a task that might prove challenging from the standpoint of convergence.

14 Cooperation and coordination with SWEET consortia and SOUR projects

Publication of the results is underway as (at least) a full research paper. The first paper including the modelization and flexibility results is being restructured after a first review and will shortly be submitted to a journal.

Communication with the greater SWEET organization and consortia was sought, e.g., at the SWEET Knowledge Transfer Seminar which I attended in March 2023 in Bern. The tool will be publicized to relevant stakeholders and some preliminary discussions were made with other SWEET participants.

References

[1] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*. Elsevier, Jan. 1994, vol. 120, pp. 157–163.

[2] L. S. Shapley, "Stochastic games," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 39, no. 10, pp. 1095–1100, Oct. 1953.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction*. MIT Press, Nov. 2018.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by Self-Play with a general reinforcement learning algorithm," Dec. 2017.

[5] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[6] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[7] L. Buşoniu, R. Babuška, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst. Man Cybern. C Appl. Rev.*, vol. 38, no. 2, pp. 156–172, 2008.

[8] H. L. Prasad, L. A. Prashanth, and S. Bhatnagar, "Actor-Critic algorithms for learning nash equilibria in n-player General-Sum games," Jan. 2014.

[9] E. Solan and N. Vieille, "Stochastic games," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 112, no. 45, pp. 13 743–13 746, Nov. 2015.

[10] J. Hu, M. P. Wellman, and Others, "Multiagent reinforcement learning: theoretical framework and an algorithm," in *ICML*, vol. 98, 1998, pp. 242–250.

[11] M. L. Littman, "Friend-or-foe q-learning in general-sum games," in *ICML*, vol. 1. researchgate.net, 2001, pp. 322–328.

[12] P. J.-J. Herings, R. J. Peeters, and Others, "Stationary equilibria in stochastic games: Structure, selection, and computation," *J. Econ. Theory*, vol. 118, no. 1, pp. 32–60, 2004.

[13] N. Akchurina, "Multiagent reinforcement learning: Algorithm converging to nash equilibrium in general-sum discounted stochastic games," *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 1, pp. 531–538, 2009.

[14] T. Börgers and R. Sarin, "Learning through reinforcement and replicator dynamics," *J. Econ. Theory*, vol. 77, no. 1, pp. 1–14, 1997.

[15] M. Geist, B. Scherrer, and O. Pietquin, "A theory of regularized Markov Decision Processes," pp. 2160–2169, Jan. 2019.

[16] E. Derman, M. Geist, and S. Mannor, “Twice regularized MDPs and the equivalence between robustness and regularization,” pp. 22 274–22 287, Oct. 2021.

[17] R. E. Funderlic and C. D. Meyer, “Sensitivity of the stationary distribution vector for an ergodic Markov chain,” *Linear Algebra Appl.*, vol. 76, pp. 1–17, Apr. 1986.

[18] M. P. Drazin, “Pseudo-Inverses in associative rings and semigroups,” *Am. Math. Mon.*, vol. 65, no. 7, pp. 506–514, Aug. 1958.

[19] Meyer, Jr., and C. D., “The condition of a finite markov chain and perturbation bounds for the limiting probabilities,” *SIAM. J. on Algebraic and Discrete Methods*, vol. 1, no. 3, pp. 273–283, Sep. 1980.

[20] S. L. Campbell, “Differentiation of the Drazin inverse,” *SIAM J. Appl. Math.*, vol. 30, no. 4, pp. 703–707, Jun. 1976.

[21] S. L. Campbell and C. D. Meyer, *Generalized Inverses of Linear Transformations*. Society for Industrial and Applied Mathematics, Jan. 2009.

[22] D. Bello and G. Riano, “Linear Programming solvers for Markov Decision Processes,” in *2006 IEEE Systems and Information Engineering Design Symposium*. ieeexplore.ieee.org, Apr. 2006, pp. 90–95.

[23] A. Waserhole, J.-P. Gayon, and V. Jost, “Linear programming formulations for queueing control problems with action decomposability,” *OSP*, Aug. 2012.

[24] C. Derman and M. Klein, “Some remarks on finite horizon markovian decision models,” *Oper. Res.*, vol. 13, no. 2, pp. 272–278, Apr. 1965.

[25] E. Altman, “Constrained markov decision processes: stochastic modeling,” https://www-sop.inria.fr/members/Eitan.Altman/PAPERS/h.pdf, accessed: 2022-12-7.

[26] Y. Yang and J. Wang, “An overview of Multi-Agent reinforcement learning from game theoretical perspective,” pp. 1–129, Nov. 2020.

[27] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, “The complexity of decentralized control of markov decision processes,” *Mathematics of OR*, vol. 27, no. 4, pp. 819–840, Nov. 2002.

[28] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun, “Approximate solutions for partially observable stochastic games with common payoffs,” in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, Jul. 2004, pp. 136–143.

[29] O. Buffet, J. Dibangoye, A. Delage, A. Saffidine, and V. Thomas, “On Bellman’s Optimality Principle for zs-POSGs,” Jun. 2020.

[30] E. A. Hansen, D. S. Bernstein, and S. Zilberstein, “Dynamic programming for partially observable stochastic games,” https://www.aaai.org/Papers/AAAI/2004/AAAI04-112.pdf, accessed: 2022-11-16.

[31] K. Horák and B. Bošanský, “Solving partially observable stochastic games with public observations,” *AAAI*, vol. 33, no. 01, pp. 2029–2036, Jul. 2019.

[32] V. Kovařík, M. Schmid, N. Burch, M. Bowling, and V. Lisý, “Rethinking formal models of partially observable multiagent decision making,” Jun. 2019.

[33] M. Šúri, T. A. Huld, and E. D. Dunlop, “PV-GIS: a web-based solar radiation database for the calculation of PV potential in europe,” *Int. J. Sustain. Energy*, vol. 24, no. 2, pp. 55–67, Jun. 2005.

[34] S. Ganzfried and T. Sandholm, “Computing equilibria in multiplayer stochastic games of imperfect information,” *Twenty-First International Joint Conference on*, pp. 140–146, 2009.

[35] U. Berger, “Brown’s original fictitious play,” *J. Econ. Theory*, vol. 135, no. 1, pp. 572–578, Jul. 2007.

[36] Y. Nesterov and A. Nemirovski, “Finding the stationary states of markov chains by iterative methods,” *Appl. Math. Comput.*, vol. 255, pp. 58–65, Mar. 2015.

[37] L. Rabiner and B. Juang, “An introduction to hidden Markov models,” *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, Jan. 1986.

[38] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: fundamental algorithms for scientific computing in python,” *Nat. Methods*, vol. 17, no. 3, pp. 261–272, Mar. 2020.

[39] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users’ guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

[40] C. Molnar, *Interpretable Machine Learning*. Lulu.com, 2020.

[41] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[42] M. Lindner, C. Aigner, R. Witzmann, F. Wirtz, I. Berber, M. Gödde, and R. Frings, “Aktuelle musternetze zur untersuchung von spannungsproblemen in der niederspannung,” in *14. Symposium Energieinnovation*, 2016.

[43] Z. Wang and R. J. Thomas, “Random topology power grid modeling and the simulation platform modeling electric power grid transmission grid : 3-phase balanced , high voltages „” 2016.

[44] G. Wang, M. Negrete-Pincetic, A. Kowli, E. Shafieepoorfard, S. Meyn, and U. V. Shanbhag, “Dynamic competitive equilibria in electricity markets,” in *Control and Optimization Methods for Electric Smart Grids*, A. Chakrabortty and M. D. Ilić, Eds. New York, NY: Springer New York, 2012, pp. 35–62.

[45] P. Huang, A. Scheller-Wolf, and K. Sycara, “Design of a multi-unit double auction e-market,” *Comput. Intell.*, vol. 18, no. 4, pp. 596–617, Nov. 2002.

Part V

Appendices

Appendix A Data of test network

In this appendix, the electrical data for the network components of the test network presented in 10.2 are reported.

#	type	from	to	l $[km]$	r $[\frac{\Omega}{km}]$	x $[\frac{\Omega}{km}]$	c $[\frac{nF}{km}]$	g $[\frac{\mu S}{km}]$	I_{max} $[kA]$
0	NAYY 4x150 SE	1	2	0.26	0.208	0.08	261	0	0.27
1	NAYY 4x50 SE	2	3	0.029	0.642	0.083	210	0	0.142
2	NAYY 4x150 SE	1	4	0.133	0.208	0.08	261	0	0.27
3	NAYY 4x150 SE	4	5	0.133	0.208	0.08	261	0	0.27
4	NAYY 4x150 SE	5	6	0.133	0.208	0.08	261	0	0.27
5	NAYY 4x150 SE	6	7	0.133	0.208	0.08	261	0	0.27
6	NAYY 4x50 SE	4	8	0.029	0.642	0.083	210	0	0.142
7	NAYY 4x50 SE	5	9	0.029	0.642	0.083	210	0	0.142
8	NAYY 4x50 SE	6	10	0.029	0.642	0.083	210	0	0.142
9	NAYY 4x50 SE	7	11	0.029	0.642	0.083	210	0	0.142
10	NAYY 4x150 SE	1	12	0.068	0.208	0.08	261	0	0.27
11	NAYY 4x150 SE	12	13	0.068	0.208	0.08	261	0	0.27
12	NAYY 4x150 SE	13	14	0.068	0.208	0.08	261	0	0.27
13	NAYY 4x150 SE	14	15	0.068	0.208	0.08	261	0	0.27
14	NAYY 4x150 SE	15	16	0.068	0.208	0.08	261	0	0.27
15	NAYY 4x120 SE	16	17	0.068	0.225	0.08	264	0	0.242
16	NAYY 4x120 SE	17	18	0.068	0.225	0.08	264	0	0.242
17	NAYY 4x50 SE	12	19	0.029	0.642	0.083	210	0	0.142
18	NAYY 4x50 SE	13	20	0.029	0.642	0.083	210	0	0.142
19	NAYY 4x50 SE	14	21	0.029	0.642	0.083	210	0	0.142
20	NAYY 4x50 SE	15	22	0.029	0.642	0.083	210	0	0.142
21	NAYY 4x50 SE	16	23	0.029	0.642	0.083	210	0	0.142
22	NAYY 4x50 SE	17	24	0.029	0.642	0.083	210	0	0.142
23	NAYY 4x50 SE	18	25	0.029	0.642	0.083	210	0	0.142

Table 6: Test network: data of lines.

Parameter	Value
hv bus	0
lv bus	1
sn [MVA]	0.16
vn hv [kV]	20
vn lv [kV]	0.4
vk %	4
vkr %	0.96
pfe [kW]	0.62
I0 %	0.31
shift °	150

Table 7: Test network: data of transformer.