# Toward Improving Solar Panel Efficiency using Reinforcement Learning

David Abel, Emily Reif, Edward C. Williams, Michael L. Littman[1]

## 1. Introduction

Solar energy offers a pollution free and sustainable means of harvesting energy directly from the sun. Considerable effort has been directed toward maximizing the efficiency of end-to-end solar systems, including the design of photovoltaic cells [15, 26], engineering new photovoltaic architectures and materials [24], and solar tracking systems [4]. Solar tracking is especially important for maximizing performance of solar panels [8, 38, 21]. Given the proper sensors and hardware, a tracking algorithm can compute the relative location of the sun in the sky throughout the day, and a controller can orient the panel to point at the sun, illustrated in Figure 1. Its goal is to minimize the angle of incidence between incoming solar radiant energy and the grid of photovoltaic cells, as in Eke and Senturk [8], Benghanem [3], King *et al.* [21] and Kalogirou [17].

Prior work has consistently demonstrated that panels using a tracking system increase the total energy by a substantial amount: Eke and Senturk [8] report that a dual-axis tracker yielded 71 kW/h, compared to a fixed panel's yield of 52 kW/h on the same day. They also report energy harvesting gains of dual-axis tracking systems over fixed systems varying from 15% to 40%, depending on the time of year. Mousazadeh *et al.* [32] report that gains from tracking can vary between 0% and 100%, while Clifford and Eastwood [5] report a gain of 23% due to tracking in simulation. Solar tracking and control result in non-trivial benefits in solar photovoltaic systems.

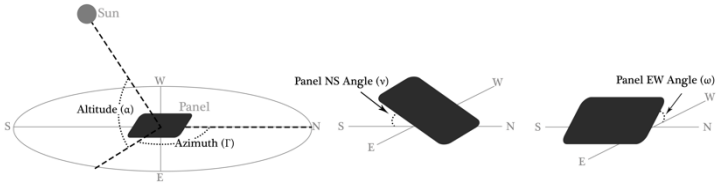



Figure 1: In the solar panel control problem, the panel changes its orientation over time to maximize total exposure to solar radiant energy.

Recent work in solar tracking has focused on creating algorithms that are sufficiently accurate to inform control of panels, building on the early work of Spencer [41], Walraven [45] and Michalsky [30]. The algorithm introduced by Reda and Andreas [37] computes the sun's location in the sky within $\pm 0.0003^{\circ}$ of accuracy, achieving the highest degree of accuracy of any known algorithm, but is computationally

[1] {david_abel, emily_reif, edward_c_williams, michael_littman}@brown.edu.
Brown University, 115 Waterman Street, Providence, RI 02912-1910

inefficient to the point of impracticality. Grena [12] overcomes these inefficiencies with a tracking algorithm that requires an order of magnitude fewer calculations while still achieving $\pm 0.0027^{\circ}$ of accuracy.

However, prior literature suggests that a variety of factors contribute to the performance of a panel [21], and thus, pointing a panel directly at the sun is not always optimal behavior. Specifically, the total solar irradiance falling on a panel is a combination of *direct*, *reflective*, and *diffuse* irradiance [3]. The diffuse irradiance typically varies between 15% and 55% of direct irradiance depending on factors like cloud coverage and the time of day [34], while a case study by the Cold Climate Housing Research Center in Fairbanks, Alaska reports reflective irradiance varying from 5% to 25% of direct irradiance [6]. The reflective irradiance varies heavily based on the percentage of irradiance reflected off the surrounding ground surface: Typical values for this percentage given by McEvoy *et al.* [29] vary between 17% (soil), 25% (grass), 55% (concrete), and 90% (snow). Additionally, changing weather and atmospheric conditions can affect the optimal panel orientation [20], as well as shading effects from neighboring panels and foliage [14, 33]. Thus, optimal performance may involve prioritizing reflective or diffuse irradiance when direct sunlight is not available.

There are several additional shortcomings to the classical tracking approach. First, tracking algorithms take as input a variety of data that require additional hardware such as a barometer, thermometer, or GPS [13], increasing the total cost and system complexity. Second, tracking algorithms are only accurate for a fixed window of time: The algorithm of Grena [12] is noted as accurate until 2023 AD (due to the subtle movements of the earth and sun), while the algorithms in Grena [13] are reported as accurate until 2110 AD. Third, trackers ignore the power cost of reorienting panels, and so waste energy on days where the benefit of movement is negligible. Lastly, shading effects from neighboring panels or foliage are ignored.

In this in progress work, we advocate for the use of Reinforcement Learning (RL) to optimize solar panel performance. A learned solar panel controller can account for weather change, cloud coverage, power use, shading effects, and diverse reflective indices of surroundings, offering an efficient yet adaptive solution that can optimize for the given availability of each type of solar irradiance without the need for complex hardware, regardless of the location or year. Our primary contribution is twofold: (1) The advancement of RL as a candidate solution for solar tracking, including an open source simulation built using recently introduced models of solar irradiance, and (2) The validation of the utility of RL approaches for solar panel control compared to standard baselines.

## 2. Background

We begin with some background on solar tracking and RL.

### 2.1 Solar Tracking

The amount of solar radiant energy contacting a surface on the earth's surface (per unit area, per unit time) is called *irradiance* [11]. We denote the total irradiance hitting a panel as $R_t$, which, per the models

developed by Kamali *et al.* [19], is approximated by the sum of the *direct* irradiance, $R_d$, *diffuse* irradiance (light from the sky), $R_f$, and *reflective* irradiance, $R_r$ (reflected off the ground or other surfaces). Each of these components is modified by a scalar, denoting the effect of the angle of incidence between oncoming solar rays and the panel's orientation, yielding the total:

$$R_t = R_d\theta_d + R_f\theta_f + R_r\theta_r$$

Additionally, the components $R_d$ and $R_f$ are known to be effected by cloud coverage [23, 35, 44].

A controller for a solar panel then seeks to maximize total irradiance, $R_t$, hitting the panel's surface. In the case of solar trackers, a running assumption is that it is near optimal to orient the panel such that its normal vector is pointing at the sun, and thus arises the necessity for accurate solar tracking algorithms. There are many types of tracking methods, only a few of which we discuss in this work; for an in-depth survey of solar tracking techniques, see Mousazadeh *et al.* [32].

**2.2 RL Background**

Reinforcement Learning is a computational learning paradigm in which an agent learns to make decisions that maximize an unknown reward function through repeated interaction with the agent's environment. In this work, we model the environment as a Markov Decision Process (MDP) [36]. An MDP is a five tuple, $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, is a reward function, $\mathcal{T}(s', a, s) = \Pr(s' \mid a, s)$, is a probability distribution on next states given a state and action, and $\gamma \in [0, 1)$, is a discount factor, indicating how much the agent prefers immediate reward over future reward. The solution to an MDP is called a *policy*, denoted $\pi : \mathcal{S} \mapsto \mathcal{A}$. The goal of an RL agent is solve for a policy that maximizes long term expected reward, defined by the *value function,* $V^*(s) : \mathcal{S} \mapsto \mathbb{R}$, defined by the classic Bellman Equation [48]:

$$V^*(s) = \max_a \left( \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s', a, s) V^*(s') \right).$$

Also of interest is the *action-value function*, $Q^* : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, which denotes the long term expected reward of executing an action in a state and behaving optimally thereafter:

$$Q^*(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s', a, s) V^*(s').$$

For further background on RL, see Sutton and Barto [42] or Kaelbling *et al.* [16].

### 2.2.1 Agents

We experiment with two standard learning algorithms: Q-Learning and SARSA, each with a linear function approximator to estimate $Q^*$. To test the significance of modeling the sequential aspects of the problem, we also conduct experiments with Q-Learning with $\gamma = 0$ (so it only maximizes immediate return) and LinUCB [25], a standard approach to non-sequential Contextual Bandits. We chose each of the linear approximators to illustrate that online, efficient, and lightweight algorithms can be effective in the domain. We chose not to experiment with any Deep RL agents [31], as Deep RL typically requires more computational power (and

often GPUs), which may be unavailable or limited in the real solar panel setting. However, this could be an area of further exploration.

Q-Learning, introduced by Watkins and Dayan [47], maintains an estimate of $Q^*$ via updates after each experience, $\langle s, a, r, s' \rangle$, updating according to the rule:

$$\hat{Q}(s,a) = (1-\eta)\hat{Q}(s,a) + \eta(r + \gamma \max_{a'} \hat{Q}(s',a')),$$

where $\eta \in [0,1]$ is a learning rate. The linear approximator extends tabular Q-Learning to domains where states are described by feature vectors, $s = [f_1, f_2, \ldots, f_k]$. Here, $\hat{Q}$ is parameterized by a set of *k*-vectors, $w_a$, where each vector corresponds to action $a$'s parameters across the state variables, resulting in:

$$\hat{Q}_w(s,a) = \sum_{i=1}^{k} w_{i,a} s_i.$$

The parameters are updated via the gradient update rule given a single experience, $\langle s, a, r, s' \rangle$:

$$w_a = w_a + \eta \left( r + \gamma \max_{a'} Q_w(s',a') - Q_w(s,a) \right).$$

SARSA [40] is similar, but makes its update using the chosen next action instead of the max action:

$$w_a = w_a + \eta \left( r + \gamma Q_w(s',a') - Q_w(s,a) \right).$$

We pair each of these algorithms with the naïve $\varepsilon$-greedy policy, which chooses actions according to:

$$\pi^{\varepsilon}(s) = \begin{cases} \arg\max_a \hat{Q}(s,a) & X = 0, X \sim \text{Bern}(\varepsilon), \\ a \sim \text{Unif}(\mathcal{A}) & otherwise, \end{cases}$$

where $\text{Unif}(\mathcal{A})$ is the uniform distribution on actions and $\text{Bern}(\varepsilon)$ is a Bernoulli distribution with parameter $\varepsilon$. To increase the expressively of the linear function approximator, we introduce some non-linearity by applying a Gaussian radial basis function to each state variable. That is, each state feature is passed through the kernel, $\phi(s_i) = e^{-(s_i^2)}$. For further details about these algorithms, see Geramifard *et al.* [9].

### 2.2.2 Contextual Bandits

A simplification of the full RL problem is a non-sequential variant known as the Contextual Bandit introduced by Wang *et al.* [46], which extends the classic Multi-Armed Bandit problem [10] to include a *context matrix, X,* containing a feature vector for each action. That is, each column of *X* corresponds to each action's context: the entry $X_{i,j}$ denotes the $i$-th feature of action $a_j$. We let $x^a$ denote the context vector associated with action $a$. At each time step, the agent chooses an action, $a \in \mathcal{A}$, and receives a payoff according to an unknown, possibly stochastic reward function, akin to the full RL problem. Here, agents still face the exploration–exploitation dilemma but do not need to learn their estimates from *delayed* reward.

We use the contextual bandit framework to assess the importance of modeling the solar panel control problem as sequential. The context vector for each action is the received percept from the environment (there is no difference in contexts across actions). Due to its simplicity and efficiency, we experiment with the LinUCB algorithm developed by Li *et al.* [25].

LinUCB adapts the core ideas of the UCB (Upper Confidence Bound) algorithm [2] to deal with contexts. At a high level, LinUCB assumes the underlying reward is determined by a linear payoff matrix $\theta$, and maintains a running estimate $\hat{\theta}$. The critical piece of the algorithm is its exploration strategy, which calculates a confidence interval on the difference between the agent's estimate of the expected reward and the actual return, which is factored into an overall *score* for each action. At each round, the agent then selects the action with maximal score according to:

$$\pi(s) = \arg\max_{a \in \mathcal{A}} \left( \sum_i \boldsymbol{s}_i^a \hat{\boldsymbol{w}}_i^a + \sigma^a \right),$$

where $\sigma^a$ represents the confidence interval associated with action a. For the full details, see Li *et al.* [25].

## 4. Experiments

A core piece of this work is the development of a simulated environment to validate the use of RL for solar panel control. There are four stages to the simulation: (1) Computing the sun's location in the sky, relative to the panel, (2) Computing $R_t$, and (3) Generating percepts. To approximate the difficulty of the real world problem, we synthesize three classes of perceptions. First, a size four vector indicating the panel's orientation and two angles that define the sun's location in the sky. The second and third classes of percepts are synthesized images of the sky – in the first case, the agent always perceives a clear sky, while in the second, we add Gaussian blobs to simulate the observational challenge posed by clouds. Sample percepts are pictured in Figure 2. Due to space constraints, we do not divulge the full details of the simulation.[2] The code for replicating results and further evaluation is publicly available.[3]

In each experiment, evaluation is done *online*, in that the agent is learning while acting to better parallel the nature of solar panel control. Our simulation is wrapped in an MDP, where the action space consists of five actions: $\mathcal{A} = \{\texttt{tilt N}, \texttt{tilt E}, \texttt{tilt S}, \texttt{tilt W}, \texttt{nothing}\}$. Executing the nothing action keeps the panel orientation fixed, while the other four each shift the panel $2^\circ$ in their respective directions. Each decision step is equivalent to three minutes of time passing.

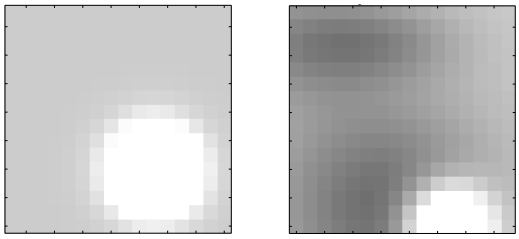


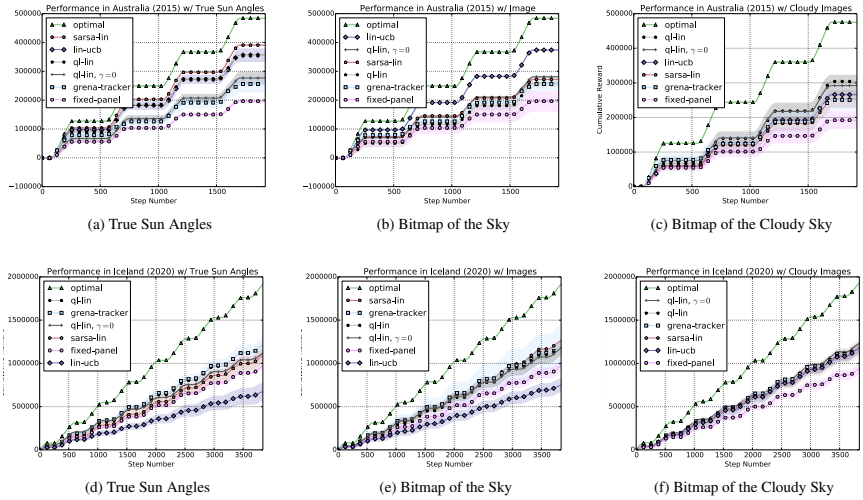
Figure 2: Example percepts given to the RL agent with no clouds (left) and simulated cloud coverage (right).

[2] For full details, see an earlier version of this work: https://cs.brown.edu/~dabel/papers/solarl.pdf
[3] https://github.com/david-abel/solar_panels_rl

Our core algorithms are Q-Learning (`ql-lin`) SARSA (`sarsa-lin`), and LinUCB (`lin-ucb`), where the input state features vary between percept types. In the first case, there are four state variables: the sun and panel angles (in radians). In the second two cases, the state variables are the pixel intensities of the image. We set exploration parameter $\varepsilon_0 = 0.3$ and learning rate $\eta_0 = 0.1$, with a standard annealing schedule, adopted from Darken and Moody [7]:

$$\eta_t = \frac{\eta_0}{\left(1.0 + \frac{t}{500}\right)}, \quad \varepsilon_t = \frac{\varepsilon_0}{\left(1.0 + \frac{t}{500}\right)}.$$

Where *t* is a time step, and the update is performed every 500 time steps. For `ql-lin` and `sarsa-lin` we chose to set $\gamma = 0.99$ to emphasize the long term consequences of behavior, contrasted with LinUCB and the short-sighted version of Q-Learning with $\gamma = 0$ (`q-lin`, $\gamma = 0$). Our core benchmark algorithm is Algorithm 2 from Grena [13], an efficient but accurate solar tracking algorithm, coupled with a controller that always points perfectly at the tracker's estimate of the sun's location. We also provide results for a fixed panel to illustrate the importance of tracking (`fixed`), and a highly idealized controller that computes the perfect orientation at each decision step to illustrate an upper bound on possible performance (`optimal`), and to visualize the degree of sub-optimality of the other approaches.




(a) True Sun Angles (b) Bitmap of the Sky (c) Bitmap of the Cloudy Sky

(d) True Sun Angles (e) Bitmap of the Sky (f) Bitmap of the Cloudy Sky

Figure 3: Cumulative irradiance falling on the panel's surface given different percepts over four days in Mildura, Australia (top) and Reykjavík, Iceland (bottom) in July of 2015

Figure 3 illustrates the cumulative irradiance exposed to each panel in simulated experiments in (top) and Iceland (bottom). Notably, with the simple percept of the true sun angles, *all* of the learning algorithms outperform the baseline tracker and fixed panel in Australia, while in Iceland, `lin-ucb` performs worse than the fixed panel for the first two percepts. When just the image is provided in Australia, `lin-ucb` achieves by far the best performance; we hypothesize that this is due to LinUCB's informed approach to exploration compared to the $\varepsilon$-greedy used by `sarsa-lin` and both `ql-lin` and `ql-lin`, $\gamma = 0$. Conversely, we see that `lin-ucb` continues to struggle in Iceland. However, when clouds are present, `lin-ucb` performs comparably to

the other learners. The cloudy image percepts pose a challenging RL problem, but still we see that the simple approaches achieve similar performance as the `grena-tracker`, and note the substantial room for improvement from further training or more complex learners. In Iceland, the results are largely the same as Australia, though we note that the `grena-tracker` does better. In all cases, there is room for improvement, suggesting that more sophisticated approaches may have more success on real panels than current techniques. In all our simulations, the `grena-tracker` consistently outperforms the fixed-panel by around 25%, consistent with previously published results [8, 32, 5].

## 5. Conclusion

We have here offered preliminary support for the use of a reinforcement-learning approach to improve the efficiency of solar panels. We offer a testbed for the problem of solar energy harvesting capable of simulating solar irradiance models with a variety of generated percepts to approximate real world conditions.

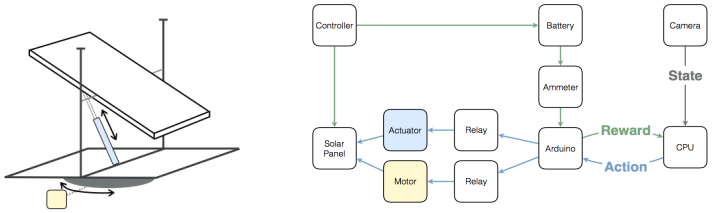



Figure 5: Panel controller prototype (left) and full system schematic diagram (right).

This work is in progress. Our immediate next step is to implement a functioning RL controller on real solar panels and compare its performance to existing approaches. We have designed a prototype, pictured in Figure 5, and are currently working on its construction. Additionally, shading, movement, observation, and computation all expend energy; incorporating these expenditures explicitly into the reinforcement learning problem poses both challenging planning and exploration questions. Lastly, in the long term, we plan on applying RL to the analogous control problem presented by solar thermal energy [18].

## References


[1] Pauli Andersen. Comments on calculations of monthly average insolation on tilted surfaces by sa klein. Solar Energy, 25(3):287, 1980.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. Machine learning, 47(2-3):235–256, 2002.

[3] M. Benghanem. Optimization of tilt angle for solar panel: Case study for Madinah, Saudi Arabia. Applied Energy, 88(4):1427–1433, 2011.

[4] Eduardo F Camacho and Manuel Berenguel. Control of solar energy systems1. IFAC Proceedings Volumes, 45(15):848–855, 2012.

[5] MJ Clifford and D Eastwood. Design of a novel passive solar tracker. Solar Energy, 77(3):269–280, 2004.

[6] Ryan Colgan, Nathan Wiltse, Michael Lilly, Ben LaRue, and Greg Egan. Performance of photovoltaic arrays. Cold Climate Housing Research Center, 2010.

[7] Christian Darken and John E Moody. Note on learning rate schedules for stochastic optimization. In NIPS, volume 91, pages 832–838, 1990.

[8] Rustu Eke and Ali Senturk. Performance comparison of a double-axis sun tracking versus fixed PV system. Solar Energy, 86(9):2665–2672, 2012.

[9] Alborz Geramifard, Thomas J Walsh, Stefanie Tellex, Girish Chowdhary, Nicholas Roy, Jonathan P How, et al. A tutorial on linear function approximators for dynamic programming and reinforcement learning. Foundations and Trends◯R in Machine Learning, 6(4):375–451, 2013.

[10] John C Gittins. Bandit processes and dynamic allocation indices. Journal of the Royal Statistical Society. Series B (Methodological), pages 148–177, 1979.

[11] D Yogi Goswami, Frank Kreith, and Jan F Kreider. Principles of solar engineering. CRC Press, 2000.

[12] Roberto Grena. An algorithm for the computation of the solar position. Solar Energy, 82(5):462–470, 2008.

[13] Roberto Grena. Five new algorithms for the computation of sun position from 2010 to 2110. Solar Energy, 86(5):1323–1337, 2012.

[14] RE Hanitsch, Detlef Schulz, and Udo Siegfried. Shading effects on output power of grid connected photovoltaic generator systems. Rev. Energ. Ren.: Power Engineering, pages 93–99, 2001.

[15] J A Jervase, H Bourdoucen, and A Al-Lawati. Solar cell parameter extraction using genetic algorithms. Measurement Science and Technology, 12(11):1922–1925, 2001.

[16] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. Journal of artificial intelligence research, 4:237–285, 1996.

[17] Soteris A Kalogirou. Design and construction of a one-axis sun-tracking system. Solar Energy, 57(6):465–469, 1996.

[18] Soteris A Kalogirou. Solar thermal collectors and applications. Progress in energy and combustion science, 30(3):231–295, 2004.

[19] Gh A Kamali, I Moradi, and A Khalili. Estimating solar radiation on tilted surfaces with various orientations: a study case in karaj (iran). Theoretical and applied climatology, 84(4):235–241, 2006.

[20] Nelson A. Kelly and Thomas L. Gibson. Improved photovoltaic energy output for cloudy conditions with a solar tracking system. Solar Energy, 83(11):2092–2102, 2009.

[21] D L King, William E Boyson, and J A Kratochvil. Analysis of factors influencing the annual energy production of photovoltaic systems. Conference Record of the Twenty-Ninth IEEE Photovoltaic Specialists Conference, 2002., pages 1356–1361, 2001.

[22] SA Klein. Calculation of monthly average insolation on tilted surfaces. Solar energy, 19(4):325–329, 1977.

[23] Danny HW Li, Chris CS Lau, and Joseph C Lam. Overcast sky conditions and luminance distribution in hong kong. Building and Environment, 39(1):101–108, 2004.

[24] Gang Li, Vishal Shrotriya, Jinsong Huang, Yan Yao, Tom Moriarty, Keith Emery, and Yang Yang. High-efficiency solution processable polymer photovoltaic cells by self-organization of polymer blends. Nature materials, 4(11):864–868, 2005.

[25] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In Proceedings of the 19th international conference on World wide web, pages 661–670. ACM, 2010.

[26] Yongfang Li. Molecular design of photovoltaic materials for polymer solar cells: toward suitable electronic energy levels and broad absorption. Accounts of Chemical Research, 45(5):723–733, 2012.

[27] Benjamin Y.H. Liu and Richard C. Jordan. The interrelationship and characteristic distribution of direct, diffuse and total solar radiation. Solar Energy, 4(3):1–19, 1960.

[28] Gilbert M Masters. Renewable and efficient electric power systems. John Wiley & Sons, 2013.

[29] Augustin McEvoy, Tom Markvart, Luis Castan˜er, T Markvart, and Luis Castaner. Practical handbook of photovoltaics: fundamentals and applications. Elsevier, 2003.

[30] Joseph J Michalsky. The astronomical almanac's algorithm for approximate solar position (1950–2050). Solar energy, 40(3):227–235, 1988.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.

[32] Hossein Mousazadeh, Alireza Keyhani, Arzhang Javadi, Hossein Mobli, Karen Abrinia, and Ahmad Sharifi. A review of principle and sun-tracking methods for maximizing solar systems output. Renewable and sustainable energy reviews, 13(8):1800–1818, 2009.

[33] Dimitrios Passias and Bengt Ka¨llba¨ck. Shading effects in rows of solar cell panels. Solar Cells, 11(3):281–291, 1984.

[34] William A Peterson and Inge Dirmhirn. The ratio of diffuse to direct solar irradiance (perpendicular to the sun's rays) with clear skies is a conserved quantity throughout the day. Journal of Applied Meteorology, 20(7):826–828, 1981.

[35] G Pfister, RL McKenzie, JB Liley, A Thomas, BW Forgan, and Charles N Long. Cloud coverage based on all-sky imaging and its impact on surface solar irradiance. Journal of Applied Meteorology, 42(10):1421–1434, 2003.

[36] Martin L Puterman. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

[37] Ibrahim Reda and Afshin Andreas. Solar position algorithm for solar radiation applications. Solar energy, 76(5):577–589, 2004.

[38] J Rizk and Y Chaiko. Solar Tracking System: More Efficient Use of Solar Panels. Proceedings of World Academy of Science: Engineering & Technology, 43:313–315, 2008.

[39] Nathan Robinson. Solar radiation. Elsevier, 1966.

[40] Gavin A Rummery and Mahesan Niranjan. On-line Q-learning using connectionist systems. University of Cambridge, Department of Engineering, 1994.

[41] JW Spencer. Fourier series representation of the position of the sun. Search, 2(5):172–172, 1971.

[42] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, volume 1. MIT press Cambridge, 1998.

[43] JL Threlkeld and RC Jordan. Direct solar radiation available on clear days. Heat., Piping Air Cond., 29(12), 1957.

[44] P Tzoumanikas, E Nikitidou, AF Bais, and A Kazantzidis. The effect of clouds on surface solar irradiance, based on data from an all-sky imaging system. Renewable Energy, 95:314–322, 2016.

[45] Robert Walraven. Calculating the position of the sun. Solar Energy, 20(5):393–397, 1978.

[46] Chih-Chun Wang, Sanjeev R Kulkarni, and H Vincent Poor. Bandit problems with side observations. IEEE Transactions on Automatic Control, 50(3):338–355, 2005.

[47] Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, 8(3):279–292, 1992.

[48] Bellman, R.E. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.