June 28, 2018

BACHELOR THESIS

OPTIMAL RACE STRATEGY FOR THE SOLAR TEAM TWENTE

Lotte Weedage

Supervisors:
dr. ir. Jasper Goseling
Jeroen Minnema

Stochastic Operations Research

1 Summary

Every two years, the Solar Team competes with their self-built Solar Car in the Bridgestone World Challenge in Australia. To win this race, it is essential to have a fast and reliable strategy. This strategy is equal to with which velocity the Solar Car needs to drive during the race and to find this optimal strategy, the race is modelled as a finite horizon Markov Decision Process. By using dynamic programming, the Markov Decision Process can be solved and an optimal solution is found. The calculated strategy can change during the race when new up-to-date data of the sun intensity is obtained. To predict the intensity of the sun during the race, the Affine Kernel Dressing method for ensemble weather predictions is used. Besides finding an optimal solution, the solutions for a risk-sensitive Markov Decision Process are constructed by using a different utility function. With this risk sensitive Markov Decision Process, the Solar Team can choose if they want to take more or less risk during the race.

Contents

2 Introduction

Every two years, the Bridgestone World Solar Challenge in Australia is held. In this challenge teams all over the world compete against each other with their Solar Car. The aim of this competition is to travel through Australia with a car that is only powered by the sun.

The Solar Team Twente has already participated in this race seven times and will do so again in 2019. For every project, they build a new car and try to improve their strategy. This strategy can also bee seen as the velocity the Solar Team will drive with during the race and this is of great importance for the Solar Team since the optimal race strategy can make the difference between winning or losing. The current race strategy of the Solar Team is deterministic: at the beginning of each day, the strategy is calculated for that day of the race, based on the weather predictions of that day.

However, it is useful to be able to change the strategy during the race if the weather differs from the prediction. During the race, a weather car drives in front of the Solar Car. This car can give weather updates to the Solar Car on desired points during the race. The wish of the Solar Team is to use these updates to possibly change their race strategy at certain times during the race and hereby decrease the finish time.

Also the implementation of risk is crucial for the Solar Team. For example, if the Solar Team is in first place, it is not necessary to take great risk, but if the team is in second place, it can be favourable to take risk and have a bigger chance of winning the race. Therefore, it would be helpful to take risk into account during the race.

In this paper, a model is introduced that represents the race of the Solar Team as a (risk sensitive) finite horizon Markov Decision Process and shows how different types of risk can change the strategy of the Solar Team. First, the formulas and approaches needed for the model, the weather predictions and the model itself is described, wherafter the implementation in Python and the results of several test runs are discussed.

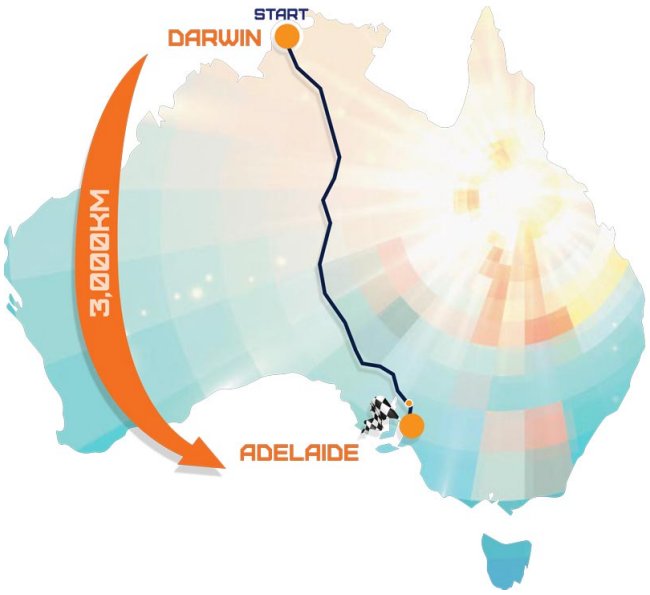

Figure 1: Race of the Solar Team in Australia

3 Problem Statement

The race strategy of the Solar Team is defined as the velocity vector that is used during the race. This velocity vector is based on the sun income and driving costs of the Solar Car. The sun income depends on the weather, since the car needs solar energy. For the weather predictions, the Solar Team uses the model of the European Meteorology Organization, ECMWF, which is provided by the KNMI. The Solar Team uses these ensemble weather predictions to predict the sun income on the different segments of the race. Depending on how much risk the Solar Team wants to take, a strategy is chosen which they will follow during that day. Risk can be seen as the chance that the Solar Car will finish the race in the predicted finish time. For example, when a big risk is taken, the probability that the car will finish in the predicted finish time is small, but this finish time is faster then when the Solar Team chooses for less risk. Therefore, the Solar Team will be advised on an implementation that can help them optimise the racing time with less or more risk.

Since a weather car is driving in front of the Solar Car, the Solar Team can use up-to-date data of the weather. This can lead to changes in the strategy during the race. To make those changes during the race, it is necessary to have a fast and accurate algorithm that can calculate new strategies during the race to minimise the finish time.

The wishes of the Solar Team lead to the following research questions:

1. How can the finish time of the Solar Team be minimised, before and during the race?
2. Is it possible to implement an efficient algorithm that calculates the strategy of the Solar Team and can this strategy be changed during the race, based on the weather updates of the weather car?
3. Given a certain risk factor, is it possible to find an optimal strategy that takes this risk into account?

4 Literature

In this section, the literature that is necessary to answer the questions is described.

There are two bachelor theses that have already looked into the strategy of the Solar Car: [1] and [2]. In the first thesis, the basis of the model that is still used is described thoroughly. In the second thesis, this model is improved by taking the weather predictions into account. For these weather predictions, ensemble weather predictions [3] are used to improve the accuracy of those predictions. However, this model was only used to validate the strategy of the Solar Team. To make a better prediction of the ensemble weather predictions, Affine Kernel Dressing [4] is used.

In the theses above, the authors use a discrete model to model the race. It is also possible to use a continuous model instead of a discrete model, as stated in [5]. In this paper, a discrete model is chosen since if needed, the segments of the race can be very small, which is very similar to continuous models. This makes the model easier to construct and probably also easier to work with.

To improve the finish time of the Solar Team, it is convenient to model the whole race and divide the race into several segments, this will elaborated more in Section 5. At each segment of the race, a decision needs to be made: with which velocity is the car going to drive for the next segment? Every decision leads to a different battery SOC and starting time for the next segment. This can be seen as a finite horizon Markov Decision Process [6]. To find the optimal strategy from this model, dynamic programming [7] is used.

Since there is a possibility that the Markov Decision Process becomes too difficult to be solved by dynamic programming, it is useful to look at different methods for solving. Therefore, it may be needed to use approximate dynamic programming, where a (sub)optimal policy can be found faster and more efficiently. There are different approaches to approximate dynamic programming: a linear programming approach [8] or looking at temporal difference learning methods [9].

For the third question, it is necessary to look into different risk functions and find out how this can be implemented in the model made for research question 1. There are several articles written about different utility functions that take risk into account, such as [10] and [11]. Those two articles use different approaches and there will be looked into which approach is most suitable to the wishes of the Solar Team.

5 Model

The race of the Solar Team during the Bridgestone World Challenge is divided as a Markov Decision Process [6] with N segments. At the start of every segment (the *decision epoch*), the velocity that the car will have during that segment is chosen. This velocity depends on the costs of driving with that velocity, the sun incomes at that segment and the battery state of charge (SOC). In this section, the approaches for calculating the driving costs and the sun income and the model for finding the optimal strategy will be discussed.

5.1 Driving costs

The driving costs when driving a certain velocity a as used by the Solar Team are stated in [2] and these are defined as follows:

$$D(a) = C(a) \cdot P(a, v_{\text{wind}}) \tag{1}$$

The time costs $C(a)$ will be elaborated further in Section 5.3.4. The power of driving with velocity a is described as the efficiency of the motor η_m times the velocity, multiplied by the sum of the drag:

$$P(a, v_{\text{wind}}) = \eta_m \cdot a \cdot F(a, v_{\text{wind}}) \tag{2}$$

$$\begin{aligned} F(a, v_{\text{wind}}) &= F_{\text{aero}}(a, v_{\text{wind}}) + F_{\text{roll}}(a, v_{\text{wind}}) \\ &= \frac{1}{2}\rho C_{DA}(a + v_{\text{wind}})^2 + mg(a \cdot C_{\text{dyn}} + C_{\text{stat}}) \end{aligned} \tag{3}$$

with:

- Air density $\rho = 1.2\text{kg/m}^3$
- Air drag coefficient times area $C_{DA} = 0.11\text{m}^2$
- Mass of the car $m = 230$ kg
- Gravitational acceleration $g = 9.81\text{m/s}^2$
- Rolling resistance (dynamic part) $C_{\text{dyn}} = -0.000113\text{s/m}$
- Rolling resistance (static part) $C_{\text{stat}} = 0.003683$

These parameters are the ones the Solar Team uses and will therefore be used in the model for now. The velocity of the wind is assumed to be zero and then, the formula for the driving costs, as derived from Formula 1 becomes:

$$D(a) = C(a) \cdot \eta_m \cdot a \cdot F(a, 0) \tag{4}$$

5.2 Sun income

To predict the sun income at a certain point during the race, weather predictions are needed. The weather predictions that are available for the Solar Team are ensemble weather predictions consisting of 50 ensemble members.

5.2.1 Ensemble weather predictions

Ensemble weather predictions are numerical weather predictions consisting of d ensemble members, also called *members*. These ensemble members give a good overview of the weather predictions and give an estimate of the uncertainty of the predictions [3]. To produce these ensemble weather predictions, the model for the numerical weather predictions is

run d times with slightly different parametrizations.

The predicted sun income of member k at time t, segment l and velocity a is defined as [2]:

$$x_k = \eta_c \cdot A_c \cdot C(a) \cdot I_k \tag{5}$$

with:

- Solar cell efficiency $\eta_c = 0.24$
- Area of the solar cells $A_c = 4\text{m}^2$
- Time costs $C(a)$ in s
- Sun intensity I_k

Since the sun intensity that is given by the ensemble weather predictions is only available every 3 hours at certain locations, the predictions need to be processed: for the sun intensity I_k at a certain point in the race, the method as described in [2] is used: by constructing a bounding box of four grid points around the desired point and then using Inverse Distance Weighting the predicted sun intensity I_k at every desired point can be calculated for each member k.

5.2.2 Affine Kernel Dressing

Since the ensemble weather predictions are not necessarily normally distributed and it is not allowed to consider the different ensemble members separately, another approach need to be found. In [4], an approach for finding a suitable distribution of ensemble members is given and this is called Affine Kernel Dressing (AKD). With AKD, it is possible to find such a distribution for the ensemble members during a certain place and time during the race. The AKD can be seen as a dressing of d different Gaussian distributions over the d ensemble weather predictions:

$$p(y : \mathbf{x}, \theta) := \frac{1}{d\sigma} \sum_{i=0}^{d-1} K\Big(\frac{y - z_i}{\sigma}\Big) \tag{6}$$

$$K(\xi) := \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\xi^2}, \tag{7}$$

where $\mathbf{x} = \{x_1, x_2, \dots x_d\}$ is the sun income of the d ensemble members at time t and segment l, with every $x \in \mathbf{x}$ equally important. Formula 6 denotes a dressing of ensemble $\mathbf{z}$, where $\mathbf{z}$ is a transformation of the ensemble $\mathbf{x}$:

$$\begin{aligned} z_i &= a x_i + r_2 m(\mathbf{x}) + r_1 \\ \sigma^2 &= h_s^2\big(s_1 + s_2 \cdot a^2 \cdot v(\mathbf{z})\big) \\ m(\mathbf{x}) &= \frac{1}{d}\sum_i x_i \\ v(\mathbf{z}) &= \frac{1}{d}\sum_i (x_i - m(\mathbf{z}))^2 \end{aligned}$$

Hereby, h_s is Silverman's factor [12] and the parameters $\theta = \{a, r_1, r_2, s_1, s_2\}$ are not known. These parameters can be found by using a test data set and comparing this data set with the predictions. However, these test data sets are not available for now and therefore, the parameters that are used in the model are $\theta = \{1, 0, 1, 0, 1\}$, which was recommended in [4].

The random variable for the sun incomes is defined as $I_l(t_l, a_l)$, with the distribution function as defined in Formula 6.

5.3 Markov Decision Process

The race is modelled as a finite horizon Markov Decision Process [6] with the following properties:

Decision epoch	$L = \{0, 1, 2, \ldots, N\}$	N: number of segments of the race
State space ($S \times T$)	$\begin{cases} S = \left\{\Delta_S \cdot i \mid i = 0, 1, \ldots, \frac{100}{\Delta_S}\right\} \\ T = \left\{\Delta_T \cdot i \mid i = 0, 1, \ldots, \frac{N \cdot \delta / a_{min}}{\Delta_T}\right\} \end{cases}$	SOC of the battery (%) Time elapsed (s)
Action space	$A = \left\{a_{min} + \Delta_A \cdot i \mid i = 0, 1, \ldots, \frac{a_{max} - a_{min}}{\Delta_A}\right\}$	Chosen velocity (m/s)
Time costs	$C(a)$	Time costs of action a (s)
Transition probability	$p(s_{l+1}, t_{l+1} \mid s_l, a_l, t_l)$	Probability to go from state (s_l, t_l) to state (s_{l+1}, t_{l+1})

The race is divided into N segments of length δ. At the beginning of segment l, defined as decision epoch l, an action a is chosen. This action has time costs $C(a)$ and affects the state space $(S \times T)$ at the next decision epoch $l+1$. These properties will be discussed later in this section. The Markov Decision Process can also be shown in a figure:

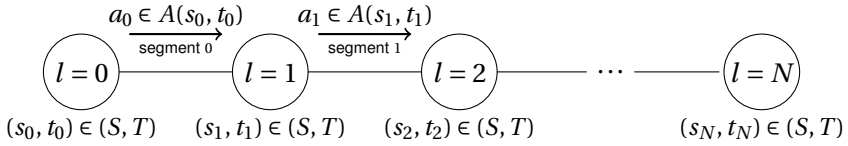

Figure 2: Markov Decision Process

5.3.1 Decision epochs

The decisions epochs in this Markov Decision Process are the points between two segments of the race. The race is evenly divided into N segments with length δ and starts at decision epoch $l = 0$. The segment before decision epoch l is the l^{th} segment, as depicted in Figure 2.

5.3.2 State space

For this model, a two-dimensional discrete state space is used. This state space $(S \times T)$ consists of the battery SOC (S) and the time (T). The battery SOC is defined as the SOC of the battery in percent at decision epoch l. Since the state space is a discrete space, the battery SOC ranges from 0 to 100 with a chosen stepsize Δ_S:

$$S = \left\{\Delta_S \cdot i \mid i = 0, 1, \ldots, \frac{100}{\Delta_S}\right\} \tag{8}$$

The same holds for the other part of the state space, the elapsed time (T) in seconds at decision epoch l. The time ranges from 0 to $N \cdot \delta / a_{min}$ with a chosen stepsize Δ_T, where a_{min} is the minimum velocity of the car:

$$T = \left\{\Delta_T \cdot i \mid i = 0, 1, \ldots, \frac{N \cdot \delta / a_{min}}{\Delta_T}\right\} \tag{9}$$

5.3.3 Action space

At every decision epoch l, the model is in a certain state (s_l, t_l). Depending on this state, an action $a_l \in A(s_l, t_l)$ is chosen. This action a_l denotes the velocity the car will have during the l^{th}

segment of the race. The set $A(s_l, t_l)$ wherein the velocity can be chosen is:

$$A(s_l, t_l) = \{a \in A | s_l + I_l(t_l, a) - D(a) > 0\} \tag{10}$$

This set of actions is a subset of the whole action space A, as defined in the Markov Decision Process. The minimum and maximum value of a and the stepsize Δ_A are chosen. The driving costs $D(a)$ and the sun income $I_l(t_l, a_l)$ are defined in Sections 5.1 and 5.2.

5.3.4 Time costs

If at a certain decision epoch l an action a_l is chosen, costs are made. These costs are defined as the time costs (in seconds) of driving with velocity a:

$$C(a) = \left\lceil \frac{\delta / a}{\Delta_T} \right\rceil \cdot \Delta_T \tag{11}$$

This function provides the value of the time costs to always be in the state space, since the state space is discrete.

5.3.5 Transition probability

The transition probability from state (s_l, t_l) to state (s_{l+1}, t_{l+1}) can be written as Formula 12, since the battery SOC and the time are independent of each other. Moreover, the probability of going from time t_l to time t_{l+1} is 1 if $t_{l+1} = t_l + C(a)$, which means that the transition probability can be written as the transition probability of s_l to s_{l+1}:

$$p(s_{l+1}, t_{l+1} \mid s_l, a_l, t_l) = p(s_{l+1} \mid s_l, a_l, t_l)\, p(t_{l+1} \mid s_l, a_l, t_l) \tag{12}$$

$$p(s_{l+1}, t_{l+1} \mid s_l, a_l, t_l) = \begin{cases} p(s_{l+1} \mid s_l, a_l, t_l), & t_{l+1} = t_l + C(a) \\ 0, & t_{l+1} \neq t_l + C(a) \end{cases} \tag{13}$$

The probability of transitioning from state s_l to state s_{l+1} is dependent on the driving costs $D(a_l)$ and the sun income $I_l(t_l, a_l)$. The sun income $I_l(t_l, a_l)$ is a random variable, which means that s_{l+1} is also a random variable:

$$s_{l+1} = \left\lfloor \frac{s_l + I_l(t_l, a_l) - D(a_l)}{\Delta_S} \right\rfloor \cdot \Delta_S \tag{14}$$

In Formula 14, the *floor*-sign is used to denote the discretisation. Since it is not necessarily true that the calculation of the new battery SOC is a value in the discrete state space battery SOC, this value is rounded down to the nearest value in this state space.

Since the probability density function of $I_l(t_l, a_l)$ is known and described in Formula 6, the transition probability will become the distribution as stated in Formula 15:

$$p(s_{l+1} | s_l, a_l, t_l) := \int_y^{y+\Delta_S} p(\hat{y} : \mathbf{x}, \theta)\, d\hat{y} \tag{15}$$

$$y = s_{l+1} - s_l + D(s_l, a_l) \tag{16}$$

5.4 Risk

As stated in the problem statement, the Solar Team wants to think about how much risk they want to take during the race, since the weather predictions are quite uncertain and differ a lot over time.

When talking about risk, there are three possibilities: risk averse, risk neutral and risk seeking. Risk averse means finding a solution with less risk: when the sun incomes turn out to be less than expected, the end time will not be a lot higher. Risk seeking, however, is the opposite: it wants to take more risk: the end time is less but also has a chance of being much more than the risk neutral and risk averse case. Risk neutral, also called risk indifferent, does not take any risk into account.

These risk preferences can be represented by a utility function over the total expected costs of the race [11]. There are different possible risk utility functions and in this case, a risk utility function for a *constant additive risk posture* is used, since the time costs of every segment are added to sum up to the final expected finish time.

The utility function for the expected time costs is defined as:

$$u(x) = \text{sgn}(\gamma) \cdot e^{\gamma x}, \tag{17}$$

where risk averse implies a negative γ and risk seeking a positive γ. While risk averse has a convex utility function, risk seeking has a concave utility function. This represents the idea of the risk sensitivity: in the risk averse case yields that the probability of getting a much higher finish time is very high. This is contrary to the risk averse case, where there is not much difference between the finish time v and $v + \Delta$.

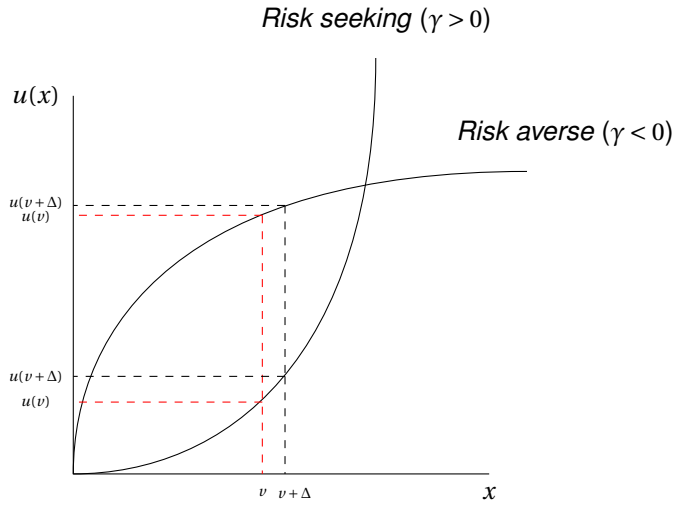

Figure 3: Convex versus concave

6 Implementation of the model

6.1 Finding the optimal velocities

The goal of this paper is to find the optimal strategy for the race. This is equal to finding a velocity vector **a** = $\{a_o, a_1, \ldots, a_{N-1}\}$ where the end time is as short as possible, taken into account that the battery SOC cannot be lower than zero. This velocity vector **a** can be found by solving Bellman's optimality equations [7]:

$$\mathbb{C}_l(s_l, t_l) = \min_{a \in A_(s_l,t_l)} \big(C(a) + \sum_{j \in S} p(j|s_l, a, t_l) \cdot \mathbb{C}_{l+1}(j, t_{l+1})\big) \tag{18}$$

$$a_l(s_l, t_l) = \operatorname*{arg\,min}_{a \in A_(s_l,t_l)} \big(C(a) + \sum_{j \in S} p(j|s_l, a, t_l) \cdot \mathbb{C}_{l+1}(j, t_{l+1})\big) \tag{19}$$

As stated in Section 5.3.5, the transition probability only depends on the battery SOC when $t_{l+1} = t_l + C(a)$, as stated in Formula's 18 and 19.

These equations are a minimisation problem over $a \in A(s_l, t_l)$. The time costs function $C(a)$ is defined in Formula 11 and $p(j|s_l, a, t_l)$ is the transition probability as defined in Formula 15. The cost function $\mathbb{C}_l(s_l, t_l)$ can be found by backwards propagation. At the last decision epoch $l = N$, the velocity (action) that is chosen is zero, since the car finished. At this decision epoch, the costs $\mathbb{C}_l(s_l, t_l)$ are zero, since the cost function resembles the time that it costs to finish.

At decision epoch $l = N - 1$, the cost $\mathbb{C}$ can be calculated by Formula 18 and this will equal the time costs of the action that is made. The cost and the corresponding decision (Formula 19) at a certain segment and state is stored in two separate matrices and is used to find the optimal solution. This continues until decision epoch $l = 0$: the start of the first segment.

When taking risk into account, the minimisation problem changes since the utility function is different. Since a exponential utility is used (Formula 17), the time costs function needs to be multiplied by the expected time costs of the next segment. Then, the optimisation equations that are used to calculate the optimal strategy become:

$$u_l(s_l, t_l) = \begin{cases} \min_{a \in A_(s_l,t_l)} \big(C(a) + \sum_{j \in S} p(j|s_l, a, t_l) \cdot \mathbb{C}_{l+1}(j, t_{l+1})\big), & \gamma = 0 \\ \min_{a \in A_(s_l,t_l)} \operatorname{sgn}(\gamma)\big(e^{\gamma C(a)} \cdot \sum_{j \in S} p(j|s_l, a, t_l) \cdot u_{l+1}(j, t_{l+1})\big)^{\gamma}, & \gamma \neq 0 \end{cases} \tag{20}$$

$$a_l(s_l, t_l) = \begin{cases} \operatorname{arg\,min}_{a \in A_(s_l,t_l)} \big(C(a) + \sum_{j \in S} p(j|s_l, a, t_l) \cdot u_{l+1}(j, t_{l+1})\big), & \gamma = 0 \\ \operatorname{arg\,min}_{a \in A_(s_l,t_l)} \operatorname{sgn}(\gamma)\big(e^{\gamma C(a)} \cdot \sum_{j \in S} p(j|s_l, a, t_l) \cdot u_{l+1}(j, t_{l+1})\big)^{\gamma}, & \gamma \neq 0 \end{cases} \tag{21}$$

6.2 Optimisation algorithm

To find the optimal velocity vector **a**, the decision matrix is used. In this decision matrix, all values of Formula 21 at all decision epochs and states are stored. The race starts at $l = 0, t = 0$ and $s = 100$. The chosen action at this segment and state can be found in the decisions matrix. With this action a, the next expected state and time can be calculated. For the calculated new battery SOC, the expected sun income is used. By using this new state $(S \times T)$, the new action a_{l+1} can be found and this algorithm repeats until the velocity of all segments is found. This is the optimal velocity vector of the race.

Moreover, the threshold 'lowest_SOC' is implemented. During the race, the Solar Team does not want to have a battery SOC lower than this 'lowest_SOC'.

6.3 Discretisation of the model

Since the Markov Decision Process uses a discrete state space, the model needs to be discrete. However, the Affine Kernel Dressing and the optimisation algorithm as described in Section 6.2 use continuous formula's. To align this and provide sensible results, everything is made discrete during the implementation of the model. This also follows from Formula 11 and 14: to make the model as realistic as possible, the battery SOC is always rounded down and the time is always rounded up.

6.4 Faster algorithm

The Solar Team wants to have a fast algorithm to make it possible to change strategies during the race. By using dynamic programming, the algorithm calculates for every state $(S \times T)$ with every action $a \in A(s_l, t_l)$ and at every segment l the utility function as stated in Formula 20. This is not necessary, however, since not all times $t \in T$ are possible. Therefore, the function is only calculated at the times that are possible during a certain segment:

$$\text{Possible time} = \left\{ \Delta_T \cdot i \mid i = \left\lfloor \frac{l \cdot \delta / a_{max}}{\Delta_T} \right\rfloor, \ldots, \left\lceil \frac{l \cdot \delta / a_{min}}{\Delta_T} \right\rceil \right\} \tag{22}$$

7 Results

To validate the model, different tests have been executed. These tests will be discussed in this section. The tests are executed with the following parameters:

- Number of segments = 10
- Segment length = 25 km
- Minimal velocity = 50 km/h
- Maximal velocity = 120 km/h
- Driving costs = 1.5 · drivingcosts
- Lowest_SOC = 0

The parameters that will change during the testing of the model are the step sizes of the battery SOC, time and actions, Δ_S, Δ_T and Δ_A. Since the tests are done with 10 segments instead of 300 segments, the driving costs are multiplied by 1.5 so the results are more interesting. The end times that are given as results are the expected end times, unless stated different, since the weather is uncertain and an expected value is given.

An example of an optimal solution is given below. The parameters used are $\Delta_A = 5, \Delta_T = 1$ and $\Delta_S = 5$. In Figure 4, a representation of the battery SOC and the actions made during the race is depicted.

```
S_optimal = [100.0, 30.0, 15.0, 0.0, 15.0, 0.0, 15.0, 0.0, 15.0, 30.0, 5.0]
A_optimal = [101.0, 76.0, 76.0, 61.0, 76.0, 61.0, 76.0, 61.0, 61.0, 64.0]
T_optimal = [0.0, 15.0, 35.0, 55.0, 80.0, 100.0, 125.0, 145.0, 170.0, 195.0, 220.0]

The race is 250 kilometers and takes 220 minutes
```

Listing 1: Optimal velocity vector

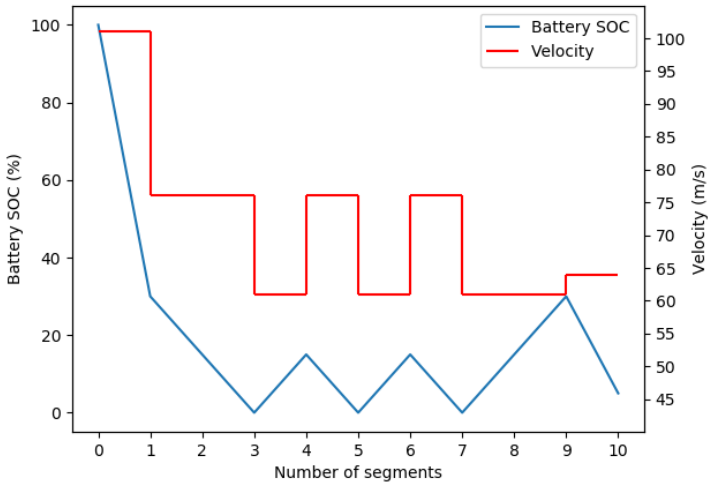

Figure 4: Battery SOC and actions

7.1 Accuracy of the model

First of all, a few runs have been done with changing some parameters to test the accuracy of the model.

Table 1: Expected finish time in minutes with $\Delta_S = 5$

Δ_A / Δ_T	**1**	**5**	**10**	**35**
1	219	224	227	245
5	220	255	270	275
10	280	280	290	300
15	300	300	300	300

When increasing the timesteps, the model becomes less accurate which can also implies higher finish times. This simulation is also illustrated in Figure 5 and 6. While the battery SOC changes quite a bit over the different simulations, the end time remains approximately the same.

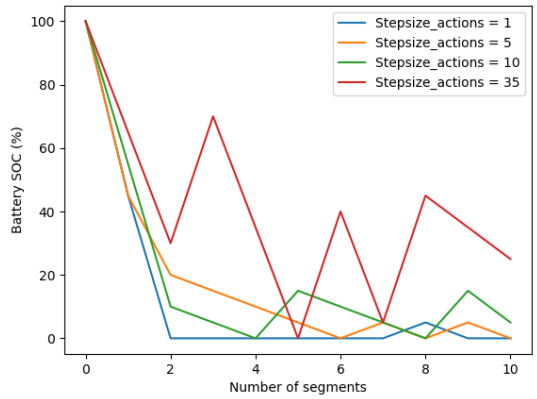

Figure 5: Battery SOC

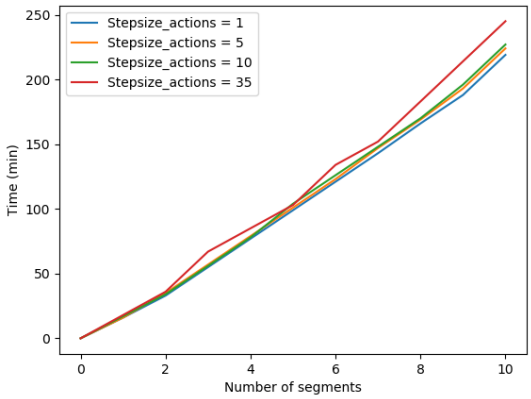

Figure 6: Race time

In these figures, the parameters that are used are: $\Delta_T = 1$, $\Delta_S = 1$ and 'lowest_SOC' = 0%.

7.2 Lowest_SOC

In the model, the parameter 'lowest_SOC' indicates the lowest battery SOC that is appropriate during the race. When this parameter changes, the battery SOC changes accordingly, as can be seen in Figure 7. Likewise, the end time changes: when the lower bound of the battery SOC goes up, so does the end time. The parameters that are used are: $\Delta_A = 1, \Delta_T = 5$ and $\Delta_S = 5$. The end times belonging to these different runs are 220, 225, 225 and 225 minutes for respectively 'lowest_SOC' = 0, 0.01, 0.05 and 0.1.

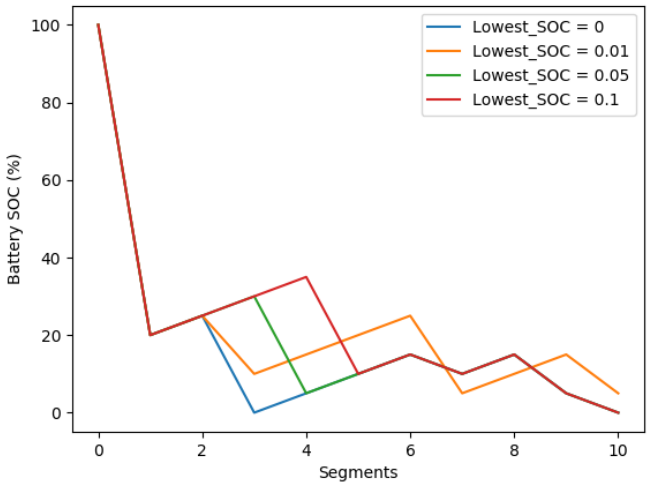

Figure 7: Changes in battery SOC with different values for lowest_SOC

7.3 Risk

To compare the different types of risk, some tests have been done. For the three different risk possibilities (risk seeking, risk neutral and risk averse, as defined in Section 5.4) the race is simulated in three different ways:

1. Sun income always less than expected
2. Sun income always higher than expected
3. Random realised sun income at every segment

The values for the risk factor γ are chosen to be $\gamma = -1$ and $\gamma = 1$ for respectively risk averse and risk seeking.

For all the three types of risk (seeking, neutral and averse), several runs have been done where the realised sun income was lower then expected, higher then expected or random. Since the run with random sun income has a large spread in the possible sun incomes, this run is done with 1000 simulations, the other two (lower and higher) have been done with 100 simulations. Table 2 shows from these combinations the minimum and maximum end time and the mean of the n simulations. The parameters that are used are: $\Delta_A = 5, \Delta_T = 5, \Delta_S = 5$ and Lowest_SOC = 0.

Table 2: min, mean and max of the finish time with n simulations

	Lower (n = 100)			**Random** (n = 1000)			**Higher** (n = 100)		
	min	mean	max	min	mean	max	min	mean	max
Averse	285	299	300	205	236	285	240	249	250
Neutral	275	293	300	205	236	285	200	201	205
Seeking	265	290	320	200	233	290	190	196	200

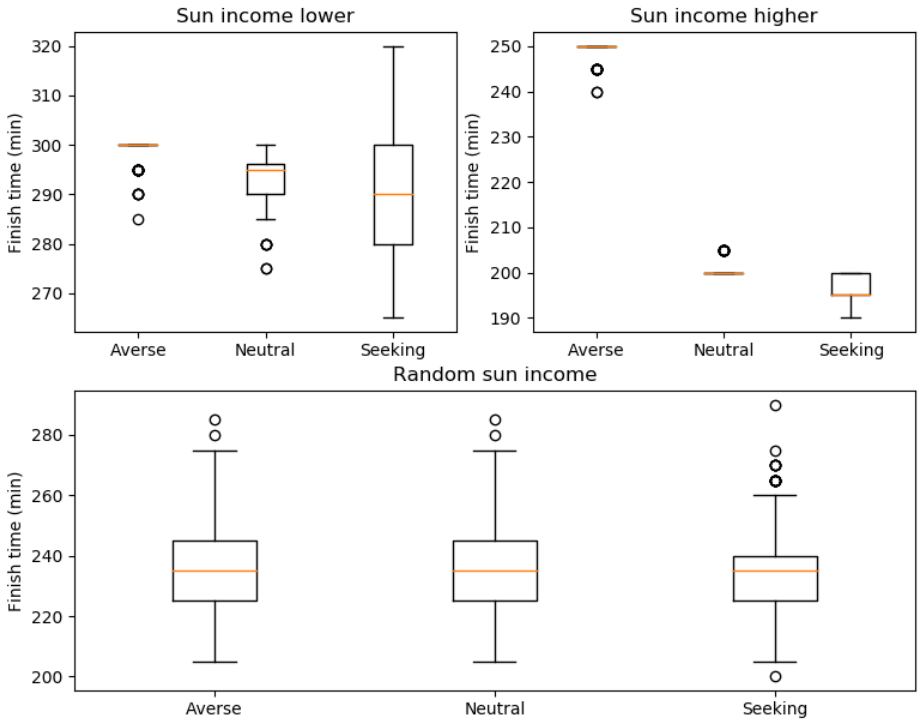

Figure 8: Box plots of the different types of risk

As depicted in Table 2 and Figure 8, the risk functions behave as expected: when the sun income is lower then expected, risk seeking gives the highest end times and the biggest variance, while risk averse gives the best finish times. When the sun income is higher then expected, risk seeking gives the best end times. Overall, risk seeking has a larger variance then risk averse. When the sun income is random, the risk averse and risk neutral case do not differ much, and risk seeking can be slightly faster.

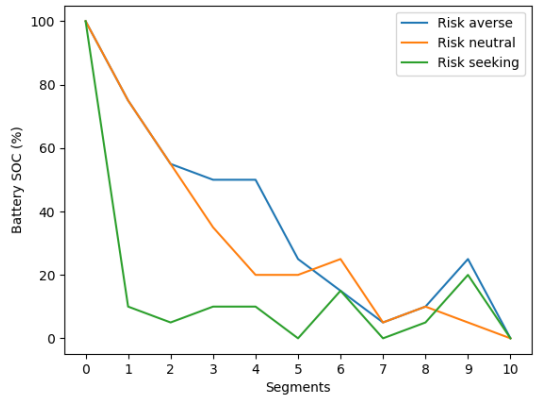

Figure 9: Lower sun income

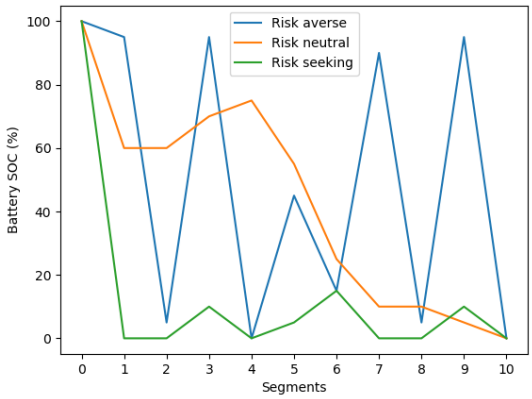

Figure 10: Higher sun income

The difference between the three risk types can also be shown in Figures 9 and 10. In these pictures, the battery SOC is showed when the sun income is lower and higher then expected. When the sun income is lower then expected, the risk averse strategy drives very carefully until the end, where it can drive with a higher velocity so the battery SOC becomes zero. Risk seeking, on the contrary, already starts with driving fast and then needs to drive slowly until the end. The neutral strategy only drives slowly, and also does not take risk at the end.

When the sun income is higher then expected, some remarkable things occur. It can be seen that risk seeking and risk neutral behaves more or less the same as the lower sun income. Risk averse, however, drives alternately fast and slow, depending on the battery SOC.

8 Conclusion

In this paper, a model for the race of the Solar Team is described and by using this model, an algorithm for finding the optimal strategy is made. This strategy gives a velocity vector which describes with which velocity the Solar Team needs to drive during the race. Since this model is discrete, the optimal strategy of the race becomes faster (lower finish time) when the step-sizes used in the model are smaller.

To implement the weather predictions in the model, the Affine Kernel Dressing method is used which gives a distribution for the weather during a certain segment and time of the race. When using the right parameters in the model, this gives an efficient algorithm that can calculate an optimal strategy. However, this implementation of the model is not yet fast and accurate enough to use during the race.

When the algorithm becomes fast enough to use during the race, the strategy can easily be changed during the race based on the sun income. Also, it is possible to take risk into account by finding an optimal strategy and change the strategy depending on the risk the Solar Team wants to take. By using a different utility function that uses a risk factor γ, the Solar Team can choose a certain risk factor to change the amount and the kind of risk they want to take. Risk averse gives a solid solution that is slower than risk seeking and risk neutral, but also changes less. Risk seeking is the opposite: it gives a fast solution with a variance and thus a bigger probability to lose.

9 Discussion

In this section, the model and implementation that is used to find the optimal strategy for the Solar Team is discussed and some recommendations are given for further research. Moreover, a number of assumptions have been made by depicting the race as a Markov Decision Process and using this model to find an optimal strategy.

- **Assumption:** In this model, the velocity of the wind in the formula for the driving costs is assumed to be zero (see Equation 4). The Solar Team has better formulas and approximations for these driving costs, but this can be implemented in the model if necessary.
- **Assumption:** As also stated in the results, some parameters and functions have been slightly changed to give interesting results and make the algorithm fast enough. This does make any changes in the behaviour of the model, this is only to show results and give the possibility to discuss these results.
- **Assumption:** Since a discrete model is used, some discretisations have been done. All discretisations concerning battery SOC have been rounded down and all discretisations concerning time have been rounded up.
- **Recommendation:** The weather predictions that are available are only available per three hours. Moreover, in the model it is assumed that the intensity during these three hours is the same. This is of course not the case, but since the sun incomes over three hours is already inaccurate, there is chosen to keep this the same. A recommendation for the Solar Team is to obtain more accurate weather predictions.
- **Recommendation:** The weather car that is driving in front of the Solar Car can give weather updates to the Solar Team. These updates can result in a better prediction of the weather which results in a better (and faster) strategy. The question is how far the weather car needs to drive in front of the Solar Car to give the best and most useful updates. This can be an important question when the strategy model as described in this thesis will be implemented as the strategy model of the Solar Team and it is recommended to look into this during further research.
- **Recommendation:** The AKD method as described in Section 5.2.2 needs proper parameters. For this model, the parameters that are used are the ones that are given in [4] as an example. For further research, it is recommended to use a training data set to find the proper parameters that belongs to these weather predictions.
- **Recommendation:** When implementing, it became clear that the model needs to be a lot more accurate to meet the needs of the Solar Team and to be useful as a model for finding the strategy during the next race. This could be done by making a faster algorithm so better parameters can be used (e.g. $\Delta_T = 1s, \Delta_S = 0.5$ and $\Delta_A = 0.1$), for example by using approximate dynamic programming [8], or by finding an analytic solution instead of a numerical solution.
  - Although the optimal solution has an error marge of $number\_segments \times \Delta_T$, this will get really big when the number of segments goes to 300. Moreover, when one looks at the finish times of previous races of the Bridgestone World Solar Challenge, it can be seen that the finish times of the first three Solar Cars does not differ more than 120 minutes, which means that the strategy needs to be a lot more accurate.
- **Recommendation:** In this model, an exponential risk utility is used. For further research, there can be looked into different risk utility functions and compare different utilities to get the best suitable risk utility function and risk factor to be the best suitable for the Solar Team.

References

[1] G. Kattenbelt, A. Goos, and T. School, “Optimale rijstrategie afhankelijk van weersvoorspellingen voor ‘The Red Engine’,” 2015.

[2] M. Siemann, “Risico van een Racestrategie voor de Red Shift,” 2017.

[3] *Guidelines on Ensemble Predictions Systems and Forecasting*, World Meteorological Organization, 2012.

[4] J. Bröcker and L. A. Smith, “From ensemble forecasts to predictive distribution functions,” *Tellus A*, vol. 60, no. 4, pp. 663–678, 2008.

[5] C. Mocking, “Optimal design and strategy for the SolUTra,” Master’s thesis, University of Twente, 2006.

[6] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[7] R. Bellman, “The theory of dynamic programming,” *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954.

[8] D. P. De Farias and B. Van Roy, “The linear programming approach to approximate dynamic programming,” *Operations research*, vol. 51, no. 6, pp. 850–865, 2003.

[9] D. P. Bertsekas and J. N. Tsitsiklis, “Neuro-dynamic programming: an overview,” in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, vol. 1. IEEE, 1995, pp. 560–564.

[10] S. C. Jaquette, “A utility criterion for Markov decision processes,” *Management Science*, vol. 23, pp. 43–49, 1976.

[11] R. A. Howard and J. E. Matheson, “Risk-sensitive Markov decision processes,” *Management science*, vol. 18, no. 7, pp. 356–369, 1972.

[12] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.