Module 1 -"Group 23"

Benjamin Pettersson 19970923-6559
(benpe@chalmers.se)
MPDSC
12 hours

January 29, 2025

We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part of other solutions. ***

***SEE THE PAGE BELOW REGARDING THE GROUP SITUATION

Please read first!

I (Benjamin Pettersson) and Hussein Joudah (joudah@chalmers.se) started this assignment the first week. We discussed the topics and sat down to write the assignment. However, during the day of submission January 29, 2025, I (Benjamin) was notified by Hussein that he had to drop out. So I was left alone in a group. I contacted Ashkan urgently about this situation. He said that he might have a partner for me for the following assignments but at the time of writing, I am doing this assignment alone. I also got permission from Hussein to submit the work we did together. Thanks for your attention.

[At time of submission]. I have not yet received an answer from Ashkan and I cannot change group on Canvas because the groups are locked. Hence, I will have to turn this assignment in and resolve the situation afterward.

Modeling alternatives considered

- Function (classification, regression, etc.).
- Optimization problem (constrained/unconstrained,...)
- Dynamic systems
- Discrete algorithmic problem
- Constraint satisfaction problem

Characteristics Considered

- Statistics and/or machine learning
- Prescriptive modeling
- Discrete algorithm design (Divide and Conquer)
- Simulation
- Optimization
- Constraint programming

Predict the Temperature

For the next weekend

When trying to predict the temperature for the next weekend, an assumption is that it is necessary to know several factors, such as: where the prediction is geographically located, what the weather conditions in the surrounding area at the current moment, and what time a year it is.

Firstly, geographic location is mentioned above since some places around the world experience faster fluctuations in day-to-day weather and will inherently be harder to predict whereas, in a place like a desert, it could be assumed that the temperature would mostly be the same if not a major fluctuation. However, the assumption is still that the problem seeks to model for the same location for the next weekend as currently.

Geographic considerations also tie into what time of year it is. As for the upper hemisphere, summer takes place from June to August, but this is flipped in the southern hemisphere. Hower, this is also a consideration negated by only considering one place, but it is an interesting note that temperature-data appears time- and place-exclusive.

Now consider the factor that is assumed to be important when modeling this problem, which is, the current weather conditions at the place considered. This is also where the thought of modeling the temperature for the next weekend as a *dynamic system* makes its appearance. Weather, and therefore the temperature, evolves based on previous states and physical laws. If there has been a heat-wave, then the air of the current area rises which in turn would cause winds when the area "refills" with as much air as previously. Hence, there are plenty of moving parts evolving during time in the system of making temperature predictions.

Modeling temperature as a dynamic system. Firstly, system variables would have to be defined, such as current temperature, humidity, wind speed, atmospheric pressure, etc. Secondly, it is necessary to introduce the dynamics of the system which is the differential equations describing the systems (easier said than done). Then it could be a necessity to set the parameters of this model which opens up a new track. How are suitable parameters for a model found? One approach is to collect data and calibrate the model from previously observed weather. Lastly, the model of a dynamic temperature system could be refined to include the previously mentioned factors that affect weather which were geographical factors, sudden weather events, and seasonal trends.

Characteristics of the model. To begin with, there is some notion of statistics lurking in the idea of forecasting (predicting) temperature. One must know how weather historically changes to conclude how weather will change. Beyond the statistical element, a dynamic system to describe the temperature would not be created for nothing but would be characterized by using simulations. The state of the current day would be data given to start the dynamic system, then it would have to be simulated forward in time to the next weekend. This is also what can be seen when visiting weather forecast websites, simulations starting at the current time and then proceeding to some time in the future. The simulations do not run for too long - since the dynamics of the system are inherently fragile. The fragility of a weather (temperature) forecast system is best described by the butterfly effect, the notion of a butterfly flapping its wings in one place of the world would end up causing a storm in some other place in the world.

Brief mention of an alternative. One alternative to predicting the temperature for the next weekend could be to model it as a function and use machine learning tied into statistics of previous years' data. I.e. consider the following mapping. Train a model to take the current temperature as input and decide how many days into the future it should try to predict. The underlying functionality

would be trained on data of previous years, if there was a similar temperature in previous years on a specific day, then it could be assumed that the future weeks weather would be somewhat similar. So the model would compare to the most similar previous years. However, there is, of course, the notion that the previous years would be "recent", i.e. 200 years back in the time might not result in the best predictive model for the current day.

Same date as today but next year

As mentioned in the previous section, weather systems (and therefore temperature) are inherently fragile. The assumption is therefore that creating a dynamic system and simulating it for the next year would be close to if not impossible. Hence, the choice of model must differ from previously used. This is why, when solving the problem of predicting the temperature for the same date as today but for the next year, alternatives must be sought after. One idea to model the problem for the next year relies on the notion that there is a need to know historical trends. Hence, to use this approach there must be data to rely on. The data must also describe the current situation, meaning it must be recent, fit the geographic situation, etc.

Before moving on to considerations of how to model the problem, one interesting note about the data is brought up. Not all years are equally long, therefore leap years exist. Hence, it could expected that data about a particular date would shift a little every year and then be reset at the leap year. This is a consideration to take into account when processing historical temperature data.

Modeling temperature prediction as a function. A general function is assumed where the input is the future date (time), historical temperatures, and external influences (such as storms). The output of this function would result in the *predictive* temperature for the same date of the next year. Several types of functions might work for this. For instance, there could be some sort of polynomial regression or even some machine learning model such as a neural network that would process the data for the previous years' temperatures.

Characteristics of the problem are statistical and/or machine learning, depending on which approach to model the problem was taken. Statistics is a necessity to describe historical temperatures. However, in this problem, taking the path of machine learning should not be ruled out given that there ought to be plenty of data to consider. For what could be considered "simple" models, regression is viewed as a form of machine learning. Therefore, statistics and machine learning define the problem.

Bingo Lottery System

In this problem, we have 1,000 tickets, 120 of which must have winning combinations. Each winning ticket should provide only one winning row, column, or diagonal. We have chosen to model the problem as a constraint satisfaction problem. The modeling also shares some characteristics of dividing and conquering, which is used in discrete algorithmic design. Hence the problem is broken down into two sub-problems: generating 120 winning tickets and 880 losing tickets, with separate modeling for each subproblem.

Winning tickets. The process begins by selecting a set of winning numbers of the 25 possible numbers. Then, the winning row, column, or diagonal is set for each winning ticket. Each ticket will contain one of three predetermined winning combinations, ensuring that no two winning tickets share identical sets of winning numbers. It becomes apparent that there are constraints involved. Every winning ticket must have at most one winning row, column, or diagonal. However, how this is achieved is not explicit. After the winning numbers have been assigned, the remaining numbers are placed in the ticket. This is to be modeled using constraint satisfaction, where variables, domain, and (implicit) constraints are defined. An example of this is such as in sudoku, where each row, column, or sub-square must only contain one of the numbers from 1 to 9. Hence, the same reasoning is used for the tickets. The model becomes:

- **Variables:** All the empty squares in the ticket are variables, however, the winning tickets receive five squares with pre-filled numbers.
- **Domain:** The domain is the numbers between 1–25. When constructing the winning tickets, the domain excludes the numbers already set.
- **Constraint:** No variable (empty square) can be directly adjacent to a square containing a winning number is one way of formulating the constraints. Another way is to form an implicit constraint saying that the tickets may not contain rows, columns, or diagonals of winning numbers.

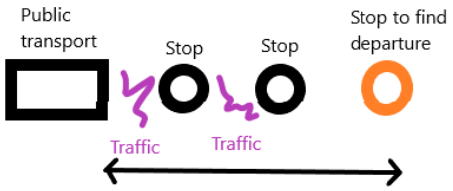
Losing tickets. For the 880 losing tickets, the process is similar but slightly modified to ensure there are no winning combinations. All 25 squares of the 5×5 grid in each ticket are treated as variables. A *Constraint Satisfaction Problem (CSP)* model is again used to populate these tickets with numbers:

- **Variables:** Each square in the ticket grid is treated as a variable, resulting in a total of 25 variables per ticket.
- **Domain:** The variables can take values from the allowed range of numbers (1–25).
- **Constraint:** The same constraint applies: no square (variable) can be directly adjacent to a square that would form part of a winning combination. Or alternatively, no diagonals, rows or columns can form a winning combination.

This approach provides a systematic method to generate unique tickets that meet the specified requirements for both winning and losing categories. The divide-and-conquer methodology characterizes the problem by breaking it into two sub-problems and further breaking the sub-problems into variables, domains, and constraints. This combined with the constraint satisfaction modeling, ensures the generation of lottery tickets.

Public Transport Departure Forecast

To address this problem, multiplte factors that could influence the outcome of the departure of public transport were considered. The major topics considered were: the distance between public transport and the considered stop, traffic conditions, time a day, and times spent at previous stops. A simplified environment is illustrated in the following Figure



The problem of modeling departure forecasts for public transport requires a breakdown into smaller measurements that could be combined into one solution. Consider, for instance, if we have information about the transport vehicles of which we wish to predict a departure, one could simply use a routing algorithm that also considers the route to the stop/station of interest. Then one could measure how long time it takes to travel the specified path as the departure time from a specific stop. However, there are more considerations.

The time it takes to travel from one place to the other is not static since traffic can vary during different times of day. Hence one would have to take into account the traffic situation given the time of day. This is where data makes its debut for the problem. A data-driven approach is necessary, using statistical models or ML to analyze traffic patterns to make predictions of how much extra time the route will take if there is much traffic. Now, both traffic and route has been accounted for. But! There are more considerations to have in mind.

Only traveling a specific route during traffic conditions does not capture the time it takes to stop at each (let us say bus) stop. People would have to get on and off the transport vehicle. Hence, data is needed here as well, in order to consider how long it takes for each stop given time of day (and which weekday). One would expect that there are more travelers during rush hour which makes for longer stops.

There could also be more factors to consider. Should a public transport departure forecast consider vehicle breakdown? Then the departure might as well become even harder to predict, i.e. when the public transport company can find an alternative substitute for the vehicle not able to departure.

In the discussion above, it is possible to argue that there are some dealings with a dynamical system as much can change and conditions are not static. However, it is not only dynamic, since a lot of factors can be accounted for given that there are statistics that describe the system. This means that one can account for disturbances from normal departures by predicting given how the environment usually fairs. Hence modelling is available using dynamic systems and functions in order to make predictions. The problem is also characterized by statistical methods, where we have to analyze the available data to estimate the time it will take for public transportation to reach some stop/station.

Film Festival Problem

There are several components to scheduling the CSE film festival. Firstly, we have a discrete amount of films, a discrete amount of persons attending, and a schedule with a set number of rooms and time slots. It is important to keep as many attendees satisfied as possible by offering an optimal movie schedule given the prioritization of the attendees. By the former information, it is argued that the problem is situated in the realm of optimization and discrete algorithms. The aim is to model a step-by-step solution that produces a "correct solution", or rather, optimal solution.

We begin by creating a model. The movies are discrete, hence let us define something such as Movies $= m_1, m_2, ..m_n$. The time slots for each movie are technically continuous, but they can be reduced into discrete slots for a specific time of one week. Hence, define the time slots as Slots $= s_1, s_2, ..., s_k$. Every attendee has a priority P_i assigned to each movie m_i. Now, we also need to decide which movie every attendee can attend. This is modeled by defining

$$x_{ik} = 0 \text{ if movie } m_i \text{ NOT in slot } s_k.\text{Else } x_{ik} = 1$$

The optimality is introduced by finding the maximum of P_i where an attendee is able to see a movie, meaning that we maximize the number of movies people can attend given the attendees personal movie ranking, the objective function is defined as

$$\text{maximizing} \sum_{i=1}^{n} \sum_{k=1}^{k} P_i \cdot x_{ik}$$

Note however that there are constraints that must be upheld. For instance, each attendee can only watch one movie at a time, which means in mathematical terms given the model that $x_{ik} + x_{kj} \leq 1$ for all j. Some constraints concern the preference of attending a certain movie, where an attendee should of course be scheduled for their high-priority movie first. Furthermore, it can be assumed that there is only one showing of a certain movie in a certain time slot at the time. Meaning that one particular movie should not run in parallel.

Notably, the problem is characterized by a constrained optimization problem. The constrained optimization problem even happens to be of discrete nature. Hence, a "mathematical programming" technique was able to be used given constraints. (Note that mathematical programming is synonymous with mathematical optimization in the field of mathematics).

The problem is also characterized by discrete algorithm design where the problems were broken down into smaller parts by dividing and conquering. In this case, broken down into variables, constraint, and an objective function.

Product Rating in Consumer Tests

Creating and determining scores for different products by users is modeling something that does not exist in reality, and therefore it is to be argued that this problem is characterized and modeled by *prescriptive modeling*. The modeling of an idea. Meaning that the intention is to create a model to quantify or define a concept that would be useful when making a decision.

Begin by considering a product. This product could be reviewed and tested in its entirety, let a consumer define a rating from 1 to 5, and settle. However, there could be more aspects to consider, so the product is broken down into smaller components on what to rate. It is argued that the characterization mentioned is a divide-and-conquer approach, solving a problem by breaking it down into smaller more manageable parts. This is also what discrete algorithm designs aim to do, dividing and conquering. However, even if it might characterize the problem, it might not be the most intuitive approach.

Consider the approach of prescriptive modeling with the characterization of smaller sub-problems, let us provide a concrete example of this. Consider creating a product rating for a hair dryer, tested by consumers. The hair dryer is not to be rated in its entirety, but components of importance. Ratings such as Price (P), Blowing Capacity (BC), and Build Quality (BQ) are considered. The ratings could be

$$\textbf{Rating} = P \cdot BC + BQ$$

or

$$\textbf{Rating} = \frac{P + BC + BQ}{\text{Total}}$$

The aim is to quantify what is considered important. The first equation finds the Price dependent on Blowing Capacity. The second does not prioritize anything special. It could also be considered to weigh the importance of the different factors by $w_1, w_2, ...$ such as

$$\textbf{Rating} = w_1 P + w_2 BC + w_3 BQ$$

Now, if the most important factor of a hair dryer was Blowing Capacity, then $w_2 > w_1, w_3$. And it could be assumed that the price is important, such as $w_2 > w_1 > w_3$. The focus will now shift from how to create the prescriptive ranking system to how to determine a score for a product in a consumer test.

There is a plurality of ways to define a score. It could be on an interval, either continuous or discrete, or possibly even a more subjective approach of using "bad" and "good". An assumption about the problem is that we have a numerical scoring. With consumer tests, we could aim to classify and predict what is thought of a certain product. I.e., this problem lays the ground for something further down the road (such as a recommender system considered in assignment 2).

Constraint Satisfaction (CS) and CS Programming

Constraint satisfaction problems (CSPs for short) can be seen as a class of computational problems that require the assignment of variables but still stay within the limits of specific constraints [1]. The constraints are there in order for the solution produced to be deemed valid. It is achieved by systematically exploring potential combinations of variable assignments, the exploration ends when the variables chosen from a domain are set successfully (adhering to the constraints).

It is possible to define constraint satisfaction problems structurally as a triple, such as [2]:

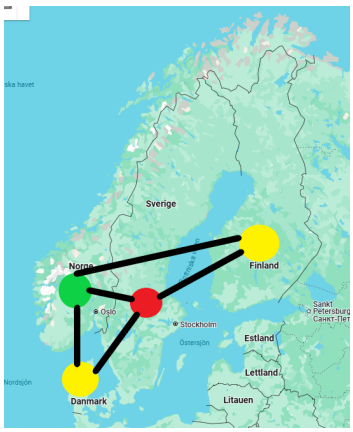
- $X = \{x_1, x_2, x_3, \dots\}$: X is a set of variables.
- $D = \{D_1, D_2, D_3, \dots\}$: D represents the domain of possible values that each variable in X can take.
- $C = \{C_1, C_2, C_3, \dots\}$: C is the set of constraints that restrict the values assignable to variables in X.

We implemented the above reasoning when the bingo lottery system was modelled. An example using sudoku was also mentioned. Let us also consider another brief example, the *map coloring problem* [3]:

Consider a map of Nordic countries. This can be modeled as a graph where the countries are vertices.

- The variables are the countries {Sweden, Finland, Denmark, Norway}.
- Set the domain of colors {Red, Yellow, Green}.
- The constraint specifies that no two neighboring countries may share the same color.

One solution for the problem is illustrated in the following Figure



Constraint Programming (CP) is a field of research where the types of problems considered are CSPs [2]. CP is then used to solve these types of combinatorial problems. I.e., variables, constraints, and domains are set, and then the CP aims to find allowed values for the variables [2]. The methodology consists of assigning allowed values to the variables and checking if constraints are satisfied, if not, then one methodology is to backtrack and try different values.

References

[1] GeeksForGeeks, "Constraint satisfaction problems (csp) in artificial intelligence." [Online]. Available: https://www.geeksforgeeks.org/constraint-satisfaction-problems-csp-in-artificial-intelligence/

[2] Wikipedia, "Constraint satisfaction problems," wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Constraint-satisfaction-problem

[3] Wikipedia Contributors, "Graph coloring," wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Graph-coloring