Ski maps

A predictive shortest-path algorithm for ski resorts

Hephaestus Applied Artificial Intelligence Association

Authors:

Member	Role
Maria Ester Massari	Head
Edoardo Ghirardo	Member
Edoardo Panella	Member
Elisa Tofanelli	Member

Milan, September 11, 2024

Contents

1 | Introduction

Skiing and snowboarding are cherished winter activities that attract adventurers to resorts across the globe. As ski resorts expand and introduce more slopes, navigating them can become increasingly intricate for tourists simply seeking to enjoy their vacation. However, in this context, Google Maps lacks effectiveness since ski lifts are typically not integrated into the mapping data. Furthermore, as of 2024, paper maps appear outdated in this digital era.
The goal of this project is to develop a tool that allows users to input their starting point and desired destination, providing instructions for the fastest route while considering both travel time and wait times at lifts. To achieve this, we'll focus on two key tools: predictive analysis and shortest path algorithms for dynamic graphs.
Predictive analysis will enable us to anticipate factors such as crowd levels, weather conditions and lift operations, allowing us to offer real-time recommendations for optimal routes. Meanwhile, shortest path algorithms will help us efficiently calculate the most time-efficient routes through the resort's slopes and ski lifts, considering the constantly changing availability of lifts and trails.
In the following pages, we'll explore how these mathematical tools can be effectively utilized to implement an efficient navigation system for ski resorts worldwide, ensuring that visitors can make the most of their time on the slopes with minimal inconvenience.

2 | Predictive analytics

The first part of the project consists in predicting the wait times at each station of the resort, based on the influencing factors. In our case, it has been very challenging to find real-life data that can be used to train a predictive model. For this reason, we will provide a theoretical background on which to work on in a (hopefully) near future when the data is available.

2.1 | Mathematical background

The data can be seen as a time-series. Mathematically, a time-series is a set of observations x, each one being recorded at a specified time t. In our case, we talk about discrete time-series since the set T_0 of times at which observations are made is a discrete set (we assume that the observations are made every minute). The theory of time-series is extensive, here we will provide a short introduction on the mathematical background used to describe them, as can be found in [3].
We can see each observation x_t as a realization of a random variable X_t. Then, we can model the data as a realization of the stochastic process $\{X_t, t \in T\}$, $T \supseteq T_0$
When analysing time-series data, we can see it as the realization of a "decomposed" model: $X_t = m_t + s_t + Y_t$, where:

- m_t is the **trend component**, a slowly changing deterministic function
- s_t is the **seasonal component**, a deterministic periodic function with known period
- Y_t is the **random noise component**.

The aim of time-series analysis is to extract the deterministic components (m_t and s_t), and find a probabilistic model for $\{Y_t\}$. Then we can use these to predict the values of X_t.
With this model in mind, various techniques exist to estimate m_t and s_t, but as at this point we are still unable to gather the data, this goes beyond the scope of this report. Please refer to [3] for further information.

2.2 | Forecasting techniques

2.2.1 | ARIMA model

Since our prediction goal is closely tied to time series forecasting, the ARIMA (Autoregressive Integrated Moving Average) model emerges as a prime candidate. It is commonly employed for time series forecasting and excels at capturing both trend and seasonality in the data, making it potentially well-suited for our application.
We will then proceed by providing a theoretical background of the ARIMA model, outlining its components and mathematical framework, referring to [1].

ARIMA (Auto-Regressive Integrated Moving Average) is a model conceptually consisting of three components:

- **AR**: the autoregressive part. Autoregression is a model that regresses a variable on its past values, for a number of lags.
- **I**: the differencing of the time series in order to make it stationary.
- **MA**: the moving average part. The moving average model regresses a variable on current and lagged error terms.

An ARIMA model is specified by 3 parameters: p is the order of the AR model, hence the number of included lags; d is the degree of differencing, that is, the number of times that the data have been substituted with first-order differences; q is the order of moving average model, similarly to p.
Thus, in the model, the **AR** component counts for the idea that the current value of the series, Xt, can be explained as a linear combination of p past values together with a random error wt, whereas the **MA** component is conceptually a linear regression of the current value of the series against current and previous white noise error terms or random shocks. Combining the **AR** and the **MA** models, without first differencing the data, is what is referred to as the family of ARMA models. A time series $\{X_t : t \in \mathbb{Z}\}$ is ARMA(p, q) if we can write

$$X_t = w_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j w_{t-j}$$

where $w_t \sim \mathcal{N}(0, \sigma_w^2)$. The ARMA model cannot be used for our purpose because the stochastic process defined by this model is stationary, while the process we want to describe is non-stationary. To address this problem, we need ways to make the data stationary, and then apply ARIMA to the transformed observations. If the root cause of non-stationarity is the presence of unit roots in the process, the data can be made stationary through differencing. This means that we difference the series, i.e. we subtract from each observation its first-lag value, as many times as unit roots there are. Applying an ARMA(p, q) on a series which has been differeced d times is effectively what is referred to as an ARIMA(p, d, q).

2.2.2 | SARIMA model

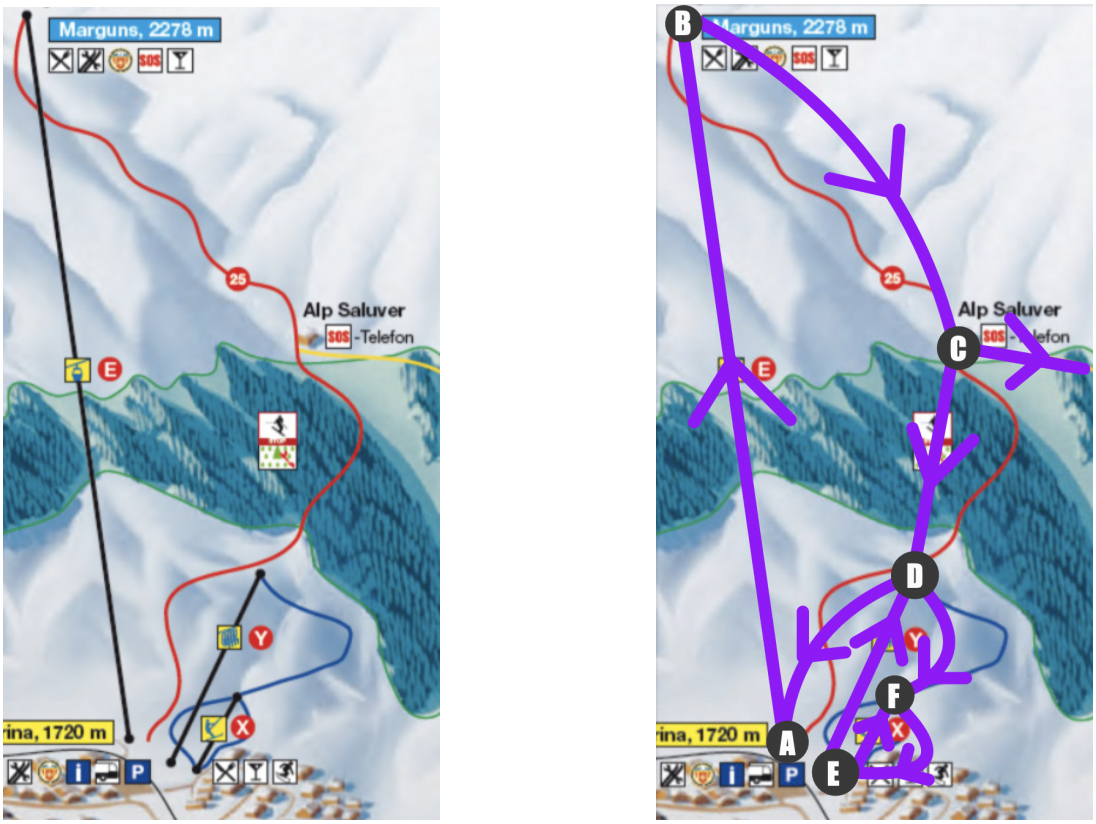
A model that would even better fit our purposes is the seasonal ARIMA model, or SARIMA, an extension of the just presented ARIMA to model even seasonal data. This model is the optimal choice for our project as it explicitly incorporates seasonal variations, crucial for accurately predicting wait times at resort stations influenced by factors such as crowd levels, weather conditions, and lift operations, ensuring robust and precise forecasts.
In particular, SARIMA includes three additional components which are basically the same ones we have seen for non-seasonal ARIMA, but with seasonal backshifts, that is, with lags being multiples of the seasonal period. This means that SARIMA is specified by 7 parameters in total: SARIMA(p, d, q)(P, D, Q)(m), where p, d and q are the non-seasonal parameters as above, P is the order of the seasonal AR component, D is the number of seasonal differences (by subtracting from the data the values at the seasonal-period-lag), Q is the order of the seasonal MA component, and m is the seasonal period. The new parameters are estimated similarly to the non-seasonal ones, and the modelling procedure in general is almost the same.

In conclusion, SARIMA modelling, tailored to capture both seasonal and non-seasonal patterns, is ideal for our project's goal of predicting wait times at resort stations. With its ability to integrate historical data and relevant factors, SARIMA ensures accurate forecasts, maximizing the effectiveness of the algorithm in optimizing operational efficiency and enhancing user experiences.

3 | Shortest path in dynamic graph

3.1 | Modelling the resort

As the goal of the project revolves around calculating traversal times within a ski resort, the most natural approach was to make use of graphs, or networks.
By assigning each departure, arrival and intersection point of slopes within a ski resort a node, and each piste and ski lift as an edge, it is possible to model its entirety in an efficient and easily manipulable format. Indeed, storing the resort as a graph significantly reduces the space utilised, and can easily be read by a program. Furthermore, extensive literature exists surrounding graph algorithms, which further backs the decision to convert maps to networks.
Specifically, each edge must be directed based on whether one can descend it using skis or ascend it in a ski lift, as shown in Figure 3.1. This intuitively represents the limitations people face when skiing; generally, it can be assumed that skiers do not ascend a piste on their skis and do not descend by chair lift. As a result, we decided to make use of a directed graph.
A further, but fundamental, consideration to make is how different pistes take different times to complete. Therefore, traversing one edge might not require the same time as traversing another. The simplest solution to this feature is to add weights to the different edges, by making use of weighted graphs.
Finally, as the waiting time will change at different moments of the day, we will adopt "dynamic" graphs, which are characterised by edges whose weight changes as a function of time.



(a) Original ski map

(b) Graph overlayed on the ski map

Figure 3.1: Graph model of a ski map

3.2 | The nature of arrivals

A further constraint given by the nature of the situation we are addressing is the fact that waiting times are defined queues, and therefore it is assumed that those who arrived first at the ski lift are those who are going to reach the top first. A similar argument is made for those skiing, as it is assumed that all participants have approximately the same ability, and therefore will arrive in the order they departed the top of the slope.
Such constraints on the order of arrivals, however, can be overcome by making use of an altered version of Dijkstra's algorithm. Indeed, while it generally would not be easily possible to find the shortest path on a dynamic graph, the First-In-First-Out condition we encountered allows for the use of arrival times in order to keep track of distances from the source edge [2].

3.3 | The algorithm

As anticipated, for the algorithm, the FIFO condition which characterizes this problem allows the use of a modified Dijkstra algorithm, as follows:

Initialization:
For all nodes i except for s, initialize $EA_{si}(t)$ as $+\infty$
Initialize $EA_{ss}(t)$ as t
Initialize S (set of unvisited nodes) with the whole N

Main loop:
While $S \neq \varnothing$
    Select $i \in S$ minimizing $EA_{si}(t)$
    Set i as visited ($S = S \setminus \{i\}$)
    For all neighbors of i (For all j such that $(i, j) \in A$):
        $EA_{sj}(t) = \min\{EA_{sj}(t), a_{ij}(EA_{si}(t))\}$

4 | Conclusion

This project demonstrates the potential of such an application of AI to the tourism sector. Indeed, while from a theoretical standpoint the model and implementation result correct according to the available literature, the major obstacle is the lack of data gathered on ski resorts. This, however, should be taken as an opportunity for further exploration with the necessary means.
Living now in an information society, ski resorts have available the resources needed to collect the necessary data, which can be enriched by that stored by meteorological stations. The biggest step to be taken now is to overcome the barrier created by data scarcity, which could be done by placing sensors on route arrivals and departures. Another possibility would be to take the anonymous location of a representative sample of volunteers, replicating on a smaller scale the approach used by larger map providers such as Google Maps and Apple Maps.
Having explored both the literature and approach utilised, what should be apparent is that this project is limited mainly by resources, rather than technological feasibility, and represents an opening in a sector which is currently mostly unexplored, especially in Europe, which calls for further exploration and expansion.
Additionally, we provide a Python implementation of the dynamic version of Dijkstra on our Github page, together with a visualization of the graph. The page offers instructions on how to use the algorithm with your own set of slopes, ski lifts and lengths.

5 | References

[1] G. M. Jenkins G. E. P. Box. *Time series analysis: Forecasting and control.* Holden-Day, 1979.

[2] Brian C. Dean. Continuous-time dynamic shortest path algorithms, 1999.

[3] Richard A. Davis Peter J. Brockwell. *Time Series: Theory and Methods, Second Edition.* Springer, 2006.