**Search algorithm:**

**Finding a path from A to destination B**

Malou Lundstrom (malou.lundstrom@students.salle.url.edu),

Lilli Zoller (lilli.zoller@students.salle.url.edu)

**Abstract:**

This paper addresses the problem of finding a path from source A to destination B using two different implementations in Python: the A\* algorithm and the Constraint Satisfaction Problem (CSP) approach. The aim is twofold: first, to implement the A\* algorithm to find the shortest path between the source and the destination, and second, to use CSP to find a path that satisfies a set of constraints. These constraints ensure the practicality of the route by limiting the total distance travelled, the duration between cities and the number of stages.

* Provide a brief summary of your paper, including the problem you addressed, your approach, key findings, and implications.

**Introduction:**

Every day there are common tasks such as GPS navigation, logistics and route planning.

This paper addresses the problem to find a path from A to a destination B. The path is a road between two cities. A provided JSON file contains the location and the connections between those cities.

To solve this problem two algorithms are used: A\* algorithm and the Constraint Satisfaction Problem (CSP) approach.

Solving this problem has significant implications and opportunities in several areas. Navigation systems with optimized routes enhance the user experience and ensure the reliability of the system. In transportation and logistics, optimized route planning can lead to cost savings and improved delivery services. The aim is to implement and compare these two approaches. By outlining the advantages and disadvantages of each, we aim to provide a comprehensive solution to the route planning problem.

* Introduce the problem you aimed to solve with your code.
* Provide background information to contextualize the problem.
* Explain the importance and relevance of solving this problem.

**Related work:**

In the field of route planning and optimization, numerous studies, YouTube videos, and papers have been dedicated to the problem of finding the optimal path.

The A\* algorithm stands out as a fundamental method for path finding. Introduced in 1968 by Nils Nilsson, Bertram Raphael and Peter Hart, the A\* algorithm was designed to find the "least-cost paths when the cost of a path is the sum of its costs" (reference). However, further research has shown the algorithm's multifaceted in finding optimal paths for a variety of problems that adhere to the conditions of cost algebra.

Reference: [https://en.wikipedia.org/wiki/A\*\_search\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm)

The Constraint Satisfaction Problem (CSP) has grown rapidly ((reference: <https://www.tuhh.de/tcs/research/constraint-satisfaction-problems>). It is a popular problem, even small problems like Sudoku, crosswords, puzzles can be modelled as a constraint statistics problem. Several different techniques have emerged in the field of CSP, including backtracking, constraint propagation, and local search. ( reference: <https://en.wikipedia.org/wiki/Constraint_satisfaction_problem>). These techniques offer different approaches to solving CSPs, each with its own strengths and limitations, making CSP a rich area of research and exploration in computer science.

Both the A\* algorithm and the Constraint Satisfaction Problem (CSP) hold a fundamental position in the field of artificial intelligence. In the AI area the A\* algorithm is most used to find the shortest path from a starting node to a destination node in a weighted graph. CSP is mostly utilized to identify and satisfy constraints within a problem domain.

(<https://www.javatpoint.com/ai-informed-search-algorithms>) + (<https://www.scaler.com/topics/courses/>)

**Methods:**

To solve this problem two algorithms are used: A\* algorithm and the Constraint Satisfaction Problem (CSP) approach.

The A\* algorithm is designed to find a solution as quickly as possible by prioritizing the expansion of nodes closest to the goal. It is not intended to find a solution at the minimum depth within the tree, but rather one that is as optimal as possible. This graph traversal and pathfinding algorithm is widely used in various areas of computer science due to its efficiency and effectiveness in minimizing travel distance or time.

On the other hand, the CSP algorithm tries to find a path by generating all optional paths and checking if they satisfy a set of constraints. These constraints are specified and include limits on the total distance travelled, the maximum duration between intermediate cities and restrictions on the number of stages in the journey. This method is particularly useful in applications such as timetabling and scheduling, where factors such as fuel capacity, time windows and driver rest periods need to be considered.

This paper builds on the foundations of these algorithms and uses their capabilities to address the task of route planning from a practical point of view. By applying and examining each approach in the context of real-world data, we aim to assess their effectiveness and suitability for different scenarios. Through this comparative analysis, we aim to highlight the relative strengths and weaknesses of each method, providing valuable insights for their application in different problem domains.

* Describe the algorithms or methods you implemented to solve the problem.
* Explain the rationale behind choosing each algorithm or method.
* Provide pseudocode or a high-level overview of your implementation.

**Data:**

The provided data is a JSON file and contains essential information for representing a map.

It consists of two main sections: cities and connections.

The Cities section provides details such as the name, address, country, latitude, and longitude of each city. These attributes help to locate the geographical location of each city.

The Connections section provides information on the distance and duration between pairs of cities which are mentioned in the Cities section. This data helps to understand the spatial relationships and travel times between different locations on the map.

To understand the dataset, we started by loading the JSON file and examining its contents. This first step gives a broad overview of the structure and attributes of the data. To get a clearer picture of the contents of the dataset, the information is printed out. This helps to understand the cities and connections present in the data. Building on this understanding, we created a graphical representation of the map. This visualization allows to see the locations of cities and the duration between them. It provides valuable insights into the layout and connectivity of the map and serves as a useful reference for our subsequent analysis and algorithm implementation.

* Describe the data you used to develop and test your code.
* Discuss any preprocessing steps you performed on the data.

**Implementation:**

* Provide details about how you implemented the code.
* Explain any design decisions, optimizations, or trade-offs you made during implementation.
* Discuss any libraries or external tools you used.

**Results:**

* Present the results of running your code on the provided data or test cases.
* Discuss the performance metrics used to evaluate your code.
* Compare your results to any existing solutions or benchmarks if available.

**Discussion:**

* Interpret and analyze the results of your code.
* Discuss the strengths and limitations of your approach.
* Address any unexpected outcomes or challenges encountered during development.

**Conclusion:**

* Summarize the key findings and contributions of your work.
* Discuss the broader implications of your results.
* Suggest potential future work or improvements.

**References:**

Have to adapt thiiiiiss

* Powerpoint -> A\* explained
* CSP explained-> https://ktiml.mff.cuni.cz/~bartak/constraints/constrsat.html

**Appendix:**

* Optionally, include additional details such as code snippets, sample input/output, or supplementary figures