Optimizing Currency Conversion Paths Using the A* Algorithm and Incorporating Volatility Rate for Enhanced Profitability

Elbert Chailes - 13522045
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
elbertchailes888@gmail.com

***Abstract*— Currency conversion is an activity conducted globally, and its cash flow significantly surpasses that of many other businesses. People worldwide engage in this activity for various reasons, including business purposes, travel, and transactions in countries with different currencies. Currency serves as a critical facilitator for easy global transactions, bridging trades and financial exchanges across international markets. In this paper, the author hypothesizes that the A* algorithm can find the optimal path for currency conversion. However, the results fail to support this hypothesis, as the A* algorithm does not consistently provide an optimal solution compared to the brute-force algorithm used as a benchmark. This method considers the volatility ratio influenced by price actions to enhance profitability.**

Keywords—currency conversion; global transactions; A* algorithm; volatility ratio; profitability

I. Introduction

Currency serves as the primary medium of exchange in economies worldwide, facilitating the trade of goods and services. It exists in various forms, ranging from physical denominations like coins and banknotes to digital formats such as electronic currencies. Given the global nature of trade and the diversity of currencies, the need for efficient currency conversion is paramount to ensure seamless economic interactions across different monetary zones.

Currency conversion is crucial as it enables transactions across countries with different currencies. This process is conducted through platforms such as money changers and forex trades, where significant volumes and price volatilities are observed. The fluctuating nature of currency values necessitates a strategic approach to conversion to avoid potential losses and maximize profitability.

This paper proposes a novel approach to optimizing currency conversion paths by employing the A* algorithm, renowned for its efficiency in finding the shortest path in various applications. By integrating considerations of conversion rates, tax implications, and market volatilities, the method aims to identify the most cost-effective routes for currency exchange. This optimization not only supports individual profitability but also enhances the overall economic efficiency of cross-border trade.

By addressing the complexities of currency conversion through an algorithmic lens, this research aims to offer a robust solution that can adapt to the dynamic nature of global finance. The use of volatility ratios and detailed tax analysis in the model ensures that the proposed solution is practical and applicable in real-world scenarios, thus resolving a common dilemma faced by many in international financial activities.

II. Limitation

While the landscape of global finance is rapidly evolving with the emergence of cryptocurrencies and digital currencies, this paper will focus primarily on the conversion of fiat currencies such as IDR, USD, JPY, among others. The decision to concentrate on fiat currencies is driven by the relatively stable factors that influence their conversion rates, unlike cryptocurrencies, which are affected by additional variables such as transaction fees ("gas fees") and higher volatility, making them less predictable and relevant for this analysis. In forex trading, where numerous currency pairs are traded, not all direct conversions are available, necessitating indirect conversion routes for many transactions. Although various methodologies could be applied to this challenge, our analysis will concentrate on comparing the A* algorithm with the brute force method. This comparison aims to highlight the efficiency of the A* algorithm against a fundamental benchmark, thereby underscoring its potential advantages in optimizing currency conversion pathways.

III. Theoretical Basis

A. *Currency Conversion*

Currency conversion is the process of exchanging one form of currency for another at an agreed exchange rate. The need for currency conversion arises from the global nature of trade, where businesses and individuals engage in transactions involving different national currencies.

Basic principles of currency conversion consisted of exchange rates, direct and indirect quotes, and cross rates. Exchange rates are rates between two currencies that specify how much one currency is worth in terms of the other which is influenced by various factors including economic indicators, market demand, and geopolitical stability. In a direct quote (D/F), the domestic currency is the base currency, whereas in an indirect quote (F/D), the domestic currency is the quote currency. To simplify it, we usually call quote currency as the target currency to convert from base currency. Cross rate is an exchange rate between two currencies calculated from their common relationships with a third currency, usually the US dollar. The formula of cross rates could be written as follows.

$$\frac{Base\ Currency}{Temp\ Currency} \times \frac{Temp\ Currency}{Quote\ Currency} = \frac{Base\ Currency}{Quote\ Currency}$$

where:

Temp Currency is a third-party currency utilized when direct exchange between a base currency and a quote currency is not available. The base currency is first exchanged into this temporary currency, which then facilitates the exchange into the desired quote currency. The exchange rates applicable in each step of this process dictate the overall effectiveness and cost of the conversion.

In currency conversion, the exchange rate is influenced by numerous external factors, making it volatile and subject to change over time. Key factors include supply and demand dynamics, market speculation, and various economic indicators. These elements can significantly affect the stability and predictability of currency values.

Additionally, there are inherent challenges associated with currency conversion. This paper categorizes these challenges as transaction costs, referred to here as "tax rates," and rate fluctuations, which we will analyze under the term "volatility rate." Both factors play crucial roles in determining the efficiency and cost-effectiveness of currency exchange processes.

B. *Volatility Rate*

Volatility rate is a statistical measure that quantifies the dispersion or variability of exchange rates around their mean over a given period. Essentially, it reflects how much and how quickly the value of one currency can change against another. Volatility is crucial for both traders and businesses that engage in international transactions. Higher volatility indicates greater risk, as the value of a currency can significantly increase or decrease in a short time.

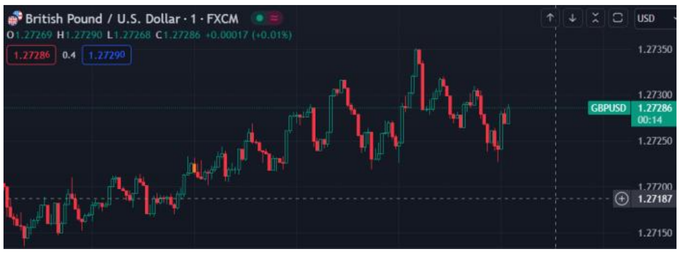


Figure 1. GBP/USD Rate Graph (1m Timeframe) (Source: www.tradingview.com)

Figure 1 illustrates the significant volatility in the GBP/USD conversion rate, highlighting frequent fluctuations that occur even within seconds. Notably, the rate changes by ±0.05% every minute on average, but spikes exceeding +1% are also possible. This variability underscores the importance of incorporating volatility rates into currency conversion calculations to mitigate potential losses and enhance profitability.

For instance, in trading scenarios where precision is crucial, even minor fluctuations can have substantial financial impacts, particularly for traders utilizing leverage. Traders aim to purchase currencies at the lowest possible cost while simultaneously managing risks associated with potential price increases that could lead to losses. Therefore, understanding and planning for volatility is critical in devising strategies that balance the pursuit of favorable rates with the need to limit exposure to rapid and unpredictable market movements.

To accurately determine the most cost-effective path for converting from one currency to another, it is crucial to consider the volatility rate. The formula for calculating annualized volatility is structured to capture these fluctuations over a standardized period. This allows for a consistent comparison and informed decision-making regarding currency exchanges. The formula is as follows:

$$Annualized\ Volatility = \sqrt{\frac{\sum(x_i - \mu)^2}{N}} \times \sqrt{252}$$

where:

X_i = daily stock price

μ = mean of prices

N = number of trading days

σ = standard deviation

The number 252 represents the typical number of trading days in a year for financial markets in the United States. This includes only the weekdays (Monday through Friday), excluding weekends and public holidays, which vary slightly from year to year and between different countries. By using 252, analysts standardize their calculations to reflect the actual number of days the market is open, and trading occurs.

C. *Tax Rate*

In currency exchange, the tax rate collectively refers to fees and service charges applied to a transaction. It does not exclusively imply government-imposed taxes but encompasses all costs incurred during the currency conversion process. The tax rate directly affects the cost-effectiveness of currency exchanges. Higher fees can significantly increase the total cost of conversion, impacting the decision-making process for businesses and individuals engaging in foreign transactions. Understanding the total tax rate applied to currency conversions is crucial for accurate financial planning and for minimizing costs in international trade and investment.

The tax rate in currency conversion is not fixed and varies depending on the trading platform, bank, or service provider facilitating the exchange. This variability means that individuals looking to convert currency should carefully select their exchange medium, prioritizing not only cost-effectiveness but also the trustworthiness of the service provider.

This variable is critical to consider because many reputable exchangers offer different tax rates for various currency pairs. Unlike exchange rates, the tax rate does not follow a predictable formula, making it an unpredictable variable in financial planning. However, it is essential to account for this in any currency conversion to ensure accurate cost assessments and avoid unexpected expenses.

Therefore, while the tax rate cannot be precisely predicted, it must be diligently researched and factored into the decision-making process when choosing a currency exchange service for a specific pair of exchange rates.

D. A Algorithm*

A* (pronounced "A-star") is a widely used graph traversal and pathfinding algorithm notable for its completeness, optimality, and efficient performance. It operates on a weighted graph to determine the shortest path between a designated source node and a goal node, based on the weights assigned to the paths.

A significant limitation of A* is its space complexity, which is $O(b^d)$, where "d" represents the depth of the shortest path and "b" is the branching factor, indicating the average number of successors per node. This complexity arises because the algorithm must keep all generated nodes in memory. Consequently, for practical travel-routing systems, A* may be less efficient than other algorithms that preprocess the graph for enhanced performance or those that use memory-bounded approaches. Despite these limitations, A* remains the preferred choice in numerous scenarios.

The algorithm was first introduced by Peter Hart, Nils Nilsson, and Bertram Raphael of the Stanford Research Institute (now SRI International) in 1968. It can be viewed as an enhancement of Dijkstra's algorithm, achieving greater efficiency by utilizing heuristics to steer its search.

Unlike Dijkstra’s algorithm, which computes the shortest paths to all potential goals from a given source, A* focuses only on finding the shortest path to a specified goal. This focus on a specific goal is made possible using goal-directed heuristics, a necessary trade-off that allows A* to be more efficient in scenarios where only a single destination matters.

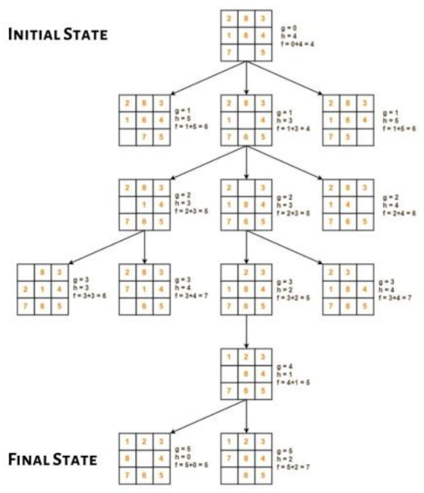


Figure 2. A* algorithm workflow example (Source: www.askpython.com)

The A* algorithm is an informed search algorithm that effectively calculates costs while exploring new nodes on the path to the solution. This algorithm utilizes two key functions to manage and estimate these costs such as *g(n)* and *h(n)*. The formula to count cost for each node could be calculated as follows.

$$f(n) = g(n) + h(n)$$

where:

- g(n) represents the actual cost from the start node to any node *n*.
- h(n) known as the heuristic function; h(n) estimates the cost from node n to the goal. This heuristic is problem-specific and is designed to help guide the search towards the goal efficiently.
- f(n) represents the estimated total cost of the cheapest solution path going through n.

Thus, the A* algorithm prioritizes processing nodes that have the lowest cost associated with *f(n)* among all active nodes. In the context of this paper, which explores currency conversion using the A* algorithm, the cost *g(n)* represents the cumulative expense incurred from the starting currency node to the current currency node. This cost can be calculated as follows.

$$g(n) = (Conversion\ Rate * Annualized\ Volatility) * (1 - tax\ rate)$$

On the other hand, *h(n)*, the heuristic component of the A* algorithm, estimates the direct conversion cost from the current currency node to the goal currency. This estimate is used even if a direct conversion pair does not exist in the current exchange platform being utilized. The heuristic is designed to be admissible, meaning it is an optimistic estimate that never overestimates the actual lowest cost *h*(n)* to reach the goal. The notation could be written that the algorithm should make sure that it fulfills $h(n) \leq h^*(n)$. This ensures that the solution provided by the A* algorithm is optimal, offering the lowest possible cost for currency conversion.

E. Brute-force Algorithm

The brute force algorithm, also known as exhaustive search, is a straightforward approach to solving computational problems by systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement. The brute force algorithm involves testing all possible solutions until the correct one is found. It does not use any shortcuts or heuristic approaches. Instead, it relies entirely on enumeration and direct evaluation. This method is commonly applied in problems with a limited and manageable number of possible solutions, such as simple puzzle games, small-scale optimization problems, or certain cryptographic attacks like password cracking.

The brute force algorithm distinguishes itself from the A* algorithm primarily because it does not consider costs or efficiencies in its search process. Instead, it systematically explores all possible solutions to determine which ones satisfy the defined boundary function. This boundary function is a crucial component that determines whether a potential solution meets the problem's requirements, classifying it as either true (valid) or false (invalid).

The time complexity of the brute force approach typically increases exponentially, often expressed as $O(2^N)$ where N represents the number of variables or elements involved. This exponential growth in complexity underscores the algorithm's inefficiency for large datasets or problems with a vast number of potential solutions.

In the context of this paper, the brute force algorithm is employed to identify all possible paths for currency conversion and to calculate the associated costs, like the *g(n)* cost function used in the A* algorithm. The primary purpose of implementing the brute force approach is to serve as a benchmark. This allows us to verify that the A* algorithm indeed provides the most optimal solution, ensuring the lowest possible conversion fees from the starting currency to the target currency.

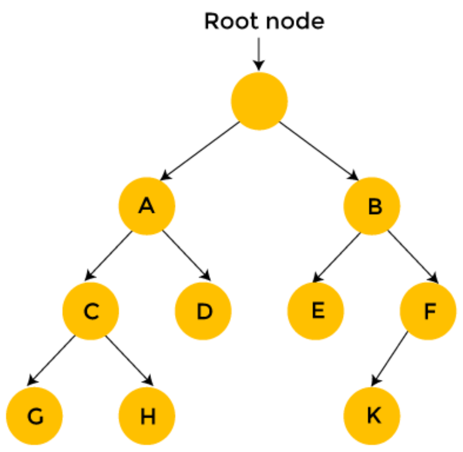


Figure 3. Brute-force graph tree illustration (Source: www.javatpoint.com)

F. *A* and Brute-force Comparison*

TABLE 1. A* VS BRUTE-FORCE COMPARISON

	A* Algorithm	Brute-force Algorithm
Pros	• A* is more efficient than brute force as it uses heuristics to guide its search, focusing on more promising paths and often reaching the solution much faster. • Better suited for larger problem sets where brute force would be computationally prohibitive.	• Brute force algorithms are straightforward to implement as they do not require any additional knowledge about the problem other than how to generate valid solutions and check them. • Brute force is guaranteed to find a solution if it exists, as it explores all possible configurations. • It always produces the correct answer, assuming enough time and computational resources, because it checks every possible option.
Cons	• A* can consume a significant amount of memory since it must store all generated nodes in its open list, which can be problematic for very large graphs or search spaces. • The performance and effectiveness of A* heavily depend on the quality of the heuristic.	• Brute force is highly inefficient for large problem spaces as it explores every possible combination without any regard for efficiency, leading to exponential time complexity. • Not suitable for problems with many variables or choices.

IV. METHODOLOGY

A. *Data Sample*

The data used in this study, particularly the exchange rate data with USD as the base currency, is sourced from ExchangeRateAPIv6's free plan. This plan limits data retrieval to rates where USD is the base, converting to various other currencies. The currency data used is static, updated according to the latest data available from the API, and does not include historical data.

To address the requirements for volatility rate analysis, volatility will be simulated with randomized values ranging from -5% to +5%. Similarly, the tax rate for currency conversion will also be randomized between 1% and 5%. These assumptions are necessary due to the absence of historical price information and aim to test the effectiveness of the algorithm in finding the most cost-effective conversion paths.

The currency pairs available for this study are selected from various exchanges, each with differing tax rates, and the volatility prices are randomized. The available pairs in this data sample will be detailed in the following tables, which serve to illustrate the scope and setup of the currency conversion paths evaluated in this research.

TABLE 2. AVAILABLE PAIRS

Base Currency	**Quote Currency**
USD	EUR, JPY, CNY, AUD
EUR	GBP, AUD, CNY
JPY	USD, EUR, CNY, GBP
GBP	USD, EUR, CHF, JPY, CAD, AUD
CHF	EUR, GBP, USD, CAD, AUD
CAD	JPY, GBP, AUD, CHF

AUD	EUR, CAD, GBP, CHF, CNY
CNY	USD, JPY, AUD, CAD

The tax rate and volatility rate are individually randomized for each currency pair as previously mentioned. For instance, consider the USD to CNY conversion where the rate is 7.25, meaning 1 USD is equivalent to 7.25 CNY. For this pair, let's assume it has a randomized tax rate of 2% and a volatility rate of 1%. Each of the other pairs in the study will also have their respective rates randomized similarly.

The dataset comprises a total of 35 possible currency pairs. It's important to note that this is just a sample, and additional currencies could be included if the API provides data for them. However, the author believes that these 35 pairs are sufficient to evaluate the efficiency and effectiveness of the algorithm in optimizing currency conversion paths.

B. *Problem Modelling*

Step 1 (Generate Graph). Based on the previously stated data sample, the algorithm constructs a graph where each node represents a base currency. The neighbors of each node represent the quote currencies available for conversion from the base currency, indicating possible conversion paths.

Step 2 (Generate Conversion Rate). While generating the graph, the algorithm calculates the cost of conversion between each adjacent currency pair. This includes adding the tax rate and volatility rate to the base conversion rate between the currencies, as outlined in the theoretical basis.

Step 3 (Input base currency and quote currency). The user inputs the base currency and the target (quote) currency, formatted as specified in the data sample (e.g., USD, JPY, GBP). If the required currency data is not in the sample, the user must manually add it to the dataset, or it should already be available via the API mentioned previously.

Step 4 (Conduct A* Search). The A* algorithm is applied to the graph, utilizing the conversion rates calculated earlier. It searches for the most cost-effective conversion path from the base currency to the quote currency.

Step 5 (Reconstruct Path). Once the A* search identifies the optimal path, the algorithm reconstructs this path into a sequential array, detailing each step of the currency conversion from base to quote currency.

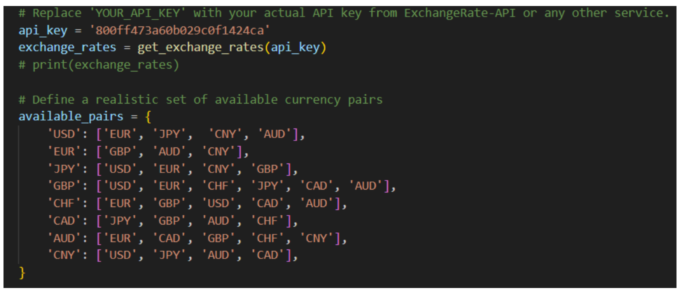
Step 6 (Output Best Conversion Path). Finally, the algorithm outputs the most optimal conversion path, including the total conversion cost accounting for taxes and volatility. This provides the user with detailed information about the most efficient way to achieve the desired currency conversion.

C. *Program Implementation to Find the Most Optimal Currency Conversion Path Considering Volatility Rate*

In the theoretical basis and problem modeling described earlier, the author has successfully implemented the optimal currency conversion process using the Python programming language. The mathematical concepts previously outlined have been translated into functional code that calculates the conversion costs from a base currency to a quote currency.

During implementation, the author developed not only the A* algorithm to find the most efficient conversion path but also utilized a brute force algorithm to evaluate all possible conversion paths. This brute force method serves as a benchmark to validate that the path identified by the A* algorithm indeed offers the lowest conversion cost.

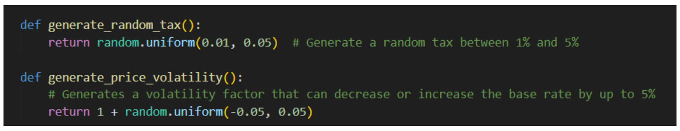
Furthermore, the author has incorporated the conversion rate data into the program, with base rates fetched from the previously mentioned API. It is important to note, however, that the data from the API may not always reflect real-time rates. Therefore, users are advised to conduct their own research and, if necessary, input their own updated data to ensure accuracy.



```python
# Replace 'YOUR_API_KEY' with your actual API key from ExchangeRate-API or any other service.
api_key = '800ff473a60b029c0f1424ca'
exchange_rates = get_exchange_rates(api_key)
# print(exchange_rates)

# Define a realistic set of available currency pairs
available_pairs = {
    'USD': ['EUR', 'JPY',  'CNY', 'AUD'],
    'EUR': ['GBP', 'AUD', 'CNY'],
    'JPY': ['USD', 'EUR', 'CNY', 'GBP'],
    'GBP': ['USD', 'EUR', 'CHF', 'JPY', 'CAD', 'AUD'],
    'CHF': ['EUR', 'GBP', 'USD', 'CAD', 'AUD'],
    'CAD': ['JPY', 'GBP', 'AUD', 'CHF'],
    'AUD': ['EUR', 'CAD', 'GBP', 'CHF', 'CNY'],
    'CNY': ['USD', 'JPY', 'AUD', 'CAD'],
}
```

Figure 4. Static available pair definition based on sample data

The author has implemented functions named generate_random_tax and generate_price_volatility to simulate tax rates and volatility. These functions use random values due to the lack of access to historical data for the currency pairs specified in the available_pairs variable. Ideally, these values would align with formulas from the theoretical basis, where volatility might mirror real market fluctuations and tax rates could reflect those of a Centralized Exchange (CEX). However, given the data constraints, it is assumed that each currency pair has its own distinct volatility and tax rate. Details of these function implementations are provided in **Figure 5.**



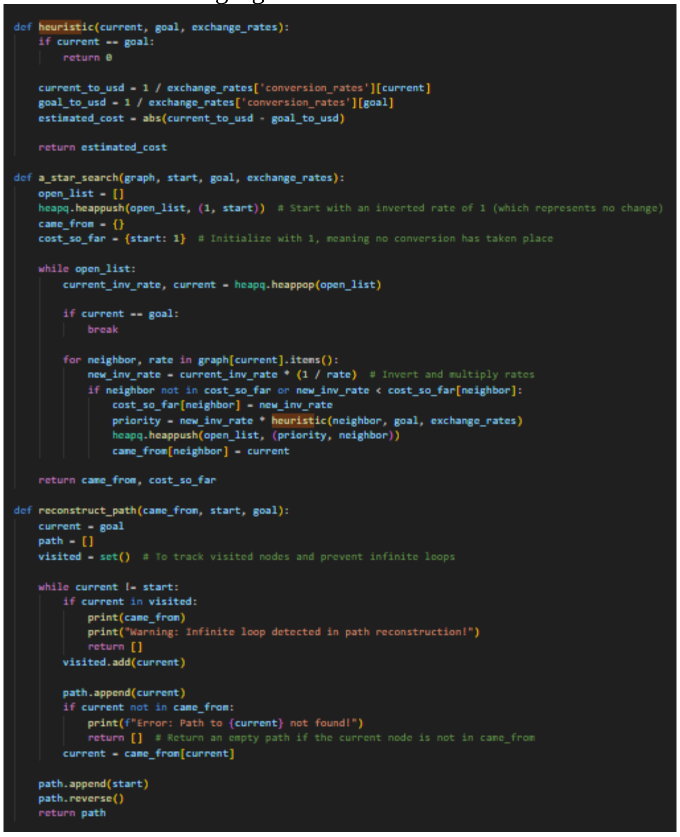
```python
def generate_random_tax():
    return random.uniform(0.01, 0.05)  # Generate a random tax between 1% and 5%

def generate_price_volatility():
    # Generates a volatility factor that can decrease or increase the base rate by up to 5%
    return 1 + random.uniform(-0.05, 0.05)
```

Figure 5. Snippet of tax and volatility rate generator

The main function orchestrates the search for the most optimal path of currency conversion, utilizing the least conversion rate. It incorporates three critical functions in the A* algorithm sequence concludes heuristic, a_star_search, and reconstruct_path.

- Heuristic Function: Developed by the author, this function implements the logic of the heuristic approach as detailed in the theoretical basis. In practice, it is translated into code to estimate the cost from a current node to the goal, aiding in the search prioritization.
- A* Search Function: Also crafted by the author, this function executes the logic for searching the most optimal path of currency conversion. It processes nodes based on the lowest f(n) value, which combines the actual path cost and the heuristic estimate.
- Reconstruct Path Function: This is a helper function that processes the output from the a_star_search function into a coherent path ready for presentation to the user. It effectively reconstructs the path from the start currency to the goal currency by tracing back through the nodes.

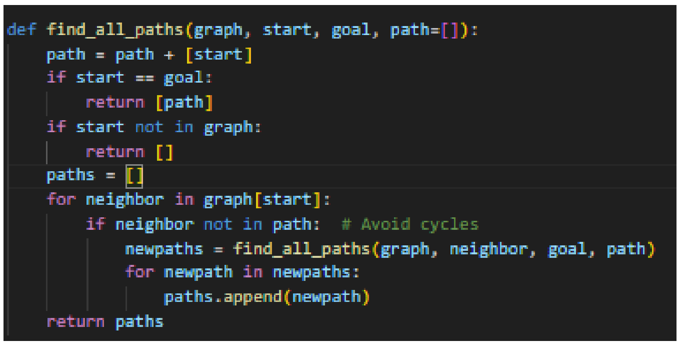
Together, these functions form the backbone of the A* algorithm applied to currency conversion, ensuring that the path with the lowest overall cost is identified and presented in an understandable format. Those 3 functions code snippet can be seen in the following figures.



```python
def heuristic(current, goal, exchange_rates):
    if current == goal:
        return 0

    current_to_usd = 1 / exchange_rates['conversion_rates'][current]
    goal_to_usd = 1 / exchange_rates['conversion_rates'][goal]
    estimated_cost = abs(current_to_usd - goal_to_usd)

    return estimated_cost

def a_star_search(graph, start, goal, exchange_rates):
    open_list = []
    heapq.heappush(open_list, (1, start))  # Start with an inverted rate of 1 (which represents no change)
    came_from = {}
    cost_so_far = {start: 1}  # Initialize with 1, meaning no conversion has taken place

    while open_list:
        current_inv_rate, current = heapq.heappop(open_list)

        if current == goal:
            break

        for neighbor, rate in graph[current].items():
            new_inv_rate = current_inv_rate * (1 / rate)  # Invert and multiply rates
            if neighbor not in cost_so_far or new_inv_rate < cost_so_far[neighbor]:
                cost_so_far[neighbor] = new_inv_rate
                priority = new_inv_rate * heuristic(neighbor, goal, exchange_rates)
                heapq.heappush(open_list, (priority, neighbor))
                came_from[neighbor] = current

    return came_from, cost_so_far

def reconstruct_path(came_from, start, goal):
    current = goal
    path = []
    visited = set()  # To track visited nodes and prevent infinite loops

    while current != start:
        if current in visited:
            print(came_from)
            print("Warning: Infinite loop detected in path reconstruction!")
            return []
        visited.add(current)

        path.append(current)
        if current not in came_from:
            print(f"Error: Path to {current} not found!")
            return []  # Return an empty path if the current node is not in came_from
        current = came_from[current]

    path.append(start)
    path.reverse()
    return path
```

Figure 6. Code snippet of A* algorithm functions

In addition to the A* algorithm, the author has implemented a brute force algorithm to serve as a benchmark. This method systematically traverses the previously constructed graph to identify all possible conversion paths, without considering costs as the A* algorithm does. The brute force approach is encapsulated in a single function, details of which are presented below.



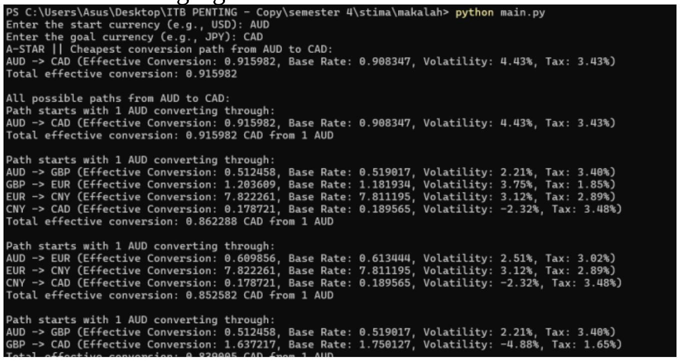
```python
def find_all_paths(graph, start, goal, path=[]):
    path = path + [start]
    if start == goal:
        return [path]
    if start not in graph:
        return []
    paths = []
    for neighbor in graph[start]:
        if neighbor not in path:  # Avoid cycles
            newpaths = find_all_paths(graph, neighbor, goal, path)
            for newpath in newpaths:
                paths.append(newpath)
    return paths
```

Figure 7. Code snippet of find_all_paths function implementation

V. Testing And Analysis

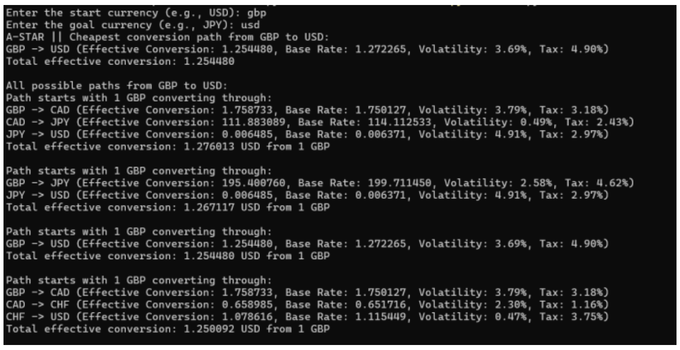
A. *Testing*

The result of the program execution after combining all the functions that could be seen in methodology chapter could be seen in following figures.



```
PS C:\Users\Asus\Desktop\ITB PENTING - Copy\semester 4\stima\makalah> python main.py
Enter the start currency (e.g., USD): AUD
Enter the goal currency (e.g., JPY): CAD
A-STAR || Cheapest conversion path from AUD to CAD:
AUD -> CAD (Effective Conversion: 0.915982, Base Rate: 0.908347, Volatility: 4.43%, Tax: 3.43%)
Total effective conversion: 0.915982

All possible paths from AUD to CAD:
Path starts with 1 AUD converting through:
AUD -> CAD (Effective Conversion: 0.915982, Base Rate: 0.908347, Volatility: 4.43%, Tax: 3.43%)
Total effective conversion: 0.915982 CAD from 1 AUD

Path starts with 1 AUD converting through:
AUD -> GBP (Effective Conversion: 0.512458, Base Rate: 0.519017, Volatility: 2.21%, Tax: 3.40%)
GBP -> EUR (Effective Conversion: 1.203609, Base Rate: 1.181934, Volatility: 3.75%, Tax: 1.85%)
EUR -> CNY (Effective Conversion: 7.822261, Base Rate: 7.811195, Volatility: 3.12%, Tax: 2.89%)
CNY -> CAD (Effective Conversion: 0.178721, Base Rate: 0.189565, Volatility: -2.32%, Tax: 3.48%)
Total effective conversion: 0.862288 CAD from 1 AUD

Path starts with 1 AUD converting through:
AUD -> EUR (Effective Conversion: 0.609856, Base Rate: 0.613444, Volatility: 2.51%, Tax: 3.02%)
EUR -> CNY (Effective Conversion: 7.822261, Base Rate: 7.811195, Volatility: 3.12%, Tax: 2.89%)
CNY -> CAD (Effective Conversion: 0.178721, Base Rate: 0.189565, Volatility: -2.32%, Tax: 3.48%)
Total effective conversion: 0.852582 CAD from 1 AUD

Path starts with 1 AUD converting through:
AUD -> GBP (Effective Conversion: 0.512458, Base Rate: 0.519017, Volatility: 2.21%, Tax: 3.40%)
GBP -> CAD (Effective Conversion: 1.637217, Base Rate: 1.750127, Volatility: -4.88%, Tax: 1.65%)
```

Figure 8. Cheapest conversion path from AUD to CAD



```
Enter the start currency (e.g., USD): gbp
Enter the goal currency (e.g., JPY): usd
A-STAR || Cheapest conversion path from GBP to USD:
GBP -> USD (Effective Conversion: 1.254480, Base Rate: 1.272265, Volatility: 3.69%, Tax: 4.90%)
Total effective conversion: 1.254480

All possible paths from GBP to USD:
Path starts with 1 GBP converting through:
GBP -> CAD (Effective Conversion: 1.758733, Base Rate: 1.750127, Volatility: 3.79%, Tax: 3.18%)
CAD -> JPY (Effective Conversion: 111.883089, Base Rate: 114.112533, Volatility: 0.49%, Tax: 2.43%)
JPY -> USD (Effective Conversion: 0.006485, Base Rate: 0.006371, Volatility: 4.91%, Tax: 2.97%)
Total effective conversion: 1.276013 USD from 1 GBP

Path starts with 1 GBP converting through:
GBP -> JPY (Effective Conversion: 195.400760, Base Rate: 199.711450, Volatility: 2.58%, Tax: 4.62%)
JPY -> USD (Effective Conversion: 0.006485, Base Rate: 0.006371, Volatility: 4.91%, Tax: 2.97%)
Total effective conversion: 1.267117 USD from 1 GBP

Path starts with 1 GBP converting through:
GBP -> USD (Effective Conversion: 1.254480, Base Rate: 1.272265, Volatility: 3.69%, Tax: 4.90%)
Total effective conversion: 1.254480 USD from 1 GBP

Path starts with 1 GBP converting through:
GBP -> CAD (Effective Conversion: 1.758733, Base Rate: 1.750127, Volatility: 3.79%, Tax: 3.18%)
CAD -> CHF (Effective Conversion: 0.658985, Base Rate: 0.651716, Volatility: 2.30%, Tax: 1.16%)
CHF -> USD (Effective Conversion: 1.078616, Base Rate: 1.115449, Volatility: 0.47%, Tax: 3.75%)
Total effective conversion: 1.250092 USD from 1 GBP
```

Figure 9. Cheapest conversion path from GBP TO USD

B. *Test Explanation*

Figure 8 presents data retrieved from the algorithmic process used to identify the most cost-effective currency conversion path from AUD to CAD. The results indicate that the optimal path, as determined by the A* algorithm, involves a direct conversion from the base currency AUD to the quote currency CAD. The conversion cost stands at 0.915982, which signifies that 1 AUD can be converted to 0.915982 CAD after accounting for the calculated tax and volatility rates, which have been randomized.

In this scenario, the tax rate for the AUD-CAD pair is 3.43%, and the volatility rate is 4.43%. The positive volatility rate suggests that there is potential for the AUD-CAD conversion rate to increase in the short term, thereby possibly offering a more profitable option momentarily. Consequently, the program recommends direct conversion from AUD to CAD as the most cost-effective path, with the specified cost.

Additionally, the brute force algorithm presents alternative conversion paths that could achieve the same result of converting from the base to the quote currency. However, these alternatives typically come with higher costs or yield less profit compared to the optimal path determined by the A* algorithm.

Figure 9 illustrates a unique case where the A* algorithm developed by the author fails to provide the most optimal path for currency conversion from the base currency to the quote currency. This is evidenced by the brute-force algorithm benchmark, which identifies the optimal conversion path from GBP to USD as GBP-CAD, CAD-JPY, and JPY-USD. This path yields a total effective conversion rate of 1.276013, meaning that 1 GBP converts to 1.276013 USD. In contrast, the A* algorithm incorrectly suggests the direct path GBP-USD, with an effective conversion rate of 1.254480, resulting in only 1.254480 USD for 1 GBP. Consequently, using the author's A* algorithm leads to a loss of 0.021533 USD compared to the brute-force benchmark.

C. *Algorithm and Method Analysis*

From the first test, shown in Figure 8, the author’s A* algorithm successfully provided an optimal path solution, matching the benchmark brute-force algorithm solution with an effective conversion cost of 0.915982. This indicates that the author’s A* algorithm has the potential to find optimal path solutions efficiently. The A* algorithm achieves this with faster time complexity since it does not visit all possible solutions but stops once it finds the first solution with the best conversion cost.

However, in the second test, depicted in Figure 9, the author’s A* algorithm failed in finding the most optimal solution. The A* algorithm found a solution with an effective conversion cost of 1.254480, while the brute-force algorithm found a better solution with an effective conversion cost of 1.276013, resulting in a loss or difference of 0.021533. The author believes this failure is due to the nature of the currency conversion problem, which involves calculating cross rates between currencies, leading to more geometric calculations rather than arithmetic ones.

The main characteristic of the A* algorithm is its use of functions such as g(n) and h(n), which sum costs and typically increase periodically with each new node visited. However, in the currency conversion problem, rates can increase and decrease dynamically. Therefore, the problem requires an algorithm that can iterate through all possible solutions rather than heuristically assuming that one solution is more effective than others based on linear cost increases. As a result, the author’s hypothesis—that the A* algorithm could consistently provide the most optimal solution—was proven false. The A* algorithm cannot always provide the most optimal solution for this problem, and there are cases where it fails.

As stated in the theoretical basis, the time complexity of the A* algorithm can be approximated by O(b^d), where d is the depth of the shortest path and b is the branching factor. This is because A* explores nodes in the order of their estimated total cost, and the number of nodes it needs to explore grows exponentially with the path length in the worst case. In contrast, the brute-force algorithm has a time complexity of O(n!) in the worst case, where each permutation of currencies is considered. This factorial time complexity arises because the algorithm generates and evaluates all possible paths, leading to an extremely high number of operations as n grows.

VI. Conclusion

Currency conversion plays a crucial role in today's interconnected world, with millions of transactions occurring every second, driven by global business activities. Given its importance, currency conversion is a vital and intriguing topic. Market participants often struggle with currency conversion due to its volatility and the fluctuating exchange rates, which can result in significant gains or losses depending on the timing of the conversion. This challenge motivated the author to explore algorithms that can efficiently find the optimal path for currency conversion using the A* algorithm.

After conducting experiments, research, and tests, the author discovered that the A* algorithm does not always provide the most optimal path for currency conversion from a base currency to a quote currency. In several cases, the brute-force algorithm, used as a benchmark, provided better optimal solutions than the A* algorithm developed by the author. Consequently, the author’s hypothesis that the A* algorithm could optimize currency conversion was proven false.

Despite this, the author successfully demonstrated that the A* algorithm has a time complexity of $O(b^d)$ in the worst case, where b is the branching factor and d is the depth of the shortest path. In contrast, the brute-force algorithm has a worst-case time complexity of $O(n!)$, where each permutation of currencies is considered. The A* algorithm struggles to find optimal solutions in scenarios that do not involve progressive cost accumulation from one node to another. Currency conversion, which aims to maximize the effective conversion rate, involves geometric calculations rather than simple arithmetic summing of costs.

In conclusion, although the A* algorithm was not able to find the optimal path for currency conversion in all cases, the brute-force algorithm succeeded in providing the most optimal solution within a manageable number of currencies (N). However, increasing the number of currencies leads to significantly longer execution times for finding the optimal path using the brute-force method.

VII. Appendix

https://github.com/ChaiGans/currency-conversion-volatility

Video Link at Youtube

https://youtu.be/eK8JJbKP-HU

ACKNOWLEDGMENT

The author extends heartfelt gratitude to the following individuals:

1. **The Almighty God**. The author acknowledges the divine blessings and guidance that have been instrumental in the successful completion of this paper. The author's faith and reliance on this support have been pivotal throughout the writing process.

2. **The Author's Parents**. The unwavering support, encouragement, and belief in the author's abilities from their parents have been invaluable. Their constant presence and motivation have served as a driving force behind the completion of this work.

3. **Dr. Ir. Rinaldi Munir, M.T.**, **Dr. Ir. Rila Mandala**, and **Dr. Nur Ulfa Maulidevi**. The author expresses deep appreciation for these esteemed professors of the IF2211 Algorithm Strategies course. Their exceptional mentorship, insightful guidance, and the extensive knowledge they imparted throughout the academic journey have significantly enriched the author's understanding of the subject matter. Their dedication and expertise have been incredibly influential.

The author recognizes and is profoundly grateful for the immeasurable contributions of these individuals, whose support and guidance have been crucial to the successful completion of this paper.

REFERENCES

[1] W. Zeng and R. L. Church, *Finding shortest paths on real road networks: the case for A**, vol. 23, no. 4, p. 531–543.

[2] "Brute force approach," javatpoint, [Online]. Available: https://www.javatpoint.com/brute-force-approach. [Accessed 12 6 2024].

[3] R. Munir, "Algoritma-Brute-Force-Bagian-1," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf. [Accessed 6 12 2024].

[4] R. Munir, "Penentuan rute (Route/Path Planning) - Bagian 1," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf. [Accessed 12 6 2024].

[5] S. J. Russell, in *Artificial intelligence a modern approach. Norvig, Peter (4th ed.).*, Boston, Pearson, 2018.

[6] D. Delling, P. Sanders, D. Schultes and D. Wagner, "Engineering Route Planning Algorithms," in *Algorithmics of Large and Complex Networks: Design, Analysis, and Simulation,* Springer, p. 117–139.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

[illegible]

Elbert Chailes (13522045)