Job Scheduling Problem Solver

Project Documentation

Project Link : https://github.com/AhmedFatthy1040/Job-Scheduling-Problem-Solver

Name	ID	Department	Level
احمد فتحى على محمد	20210089	CS	3
احمد مشرف محمد	20210120	CS	3
ياسر شبير محمد	20211042	AI	3
روان محمد	20210342	AI	3
روان محمود	20210343	AI	3
روان خالد	20210339	AI	3
شروق محمد سالم	20210450	AI	3

Project Idea: Job Scheduling Optimization System with Backtracking and Genetic Algorithms

Overview:

This project focuses on developing a comprehensive job scheduling optimization system that leverages both Backtracking and Genetic Algorithms. The goal is to provide users with effective solutions to the Job Scheduling Problem, offering insights into the strengths and limitations of each algorithm. The system will enable users to make informed decisions when selecting approaches for specific instances of the problem, contributing to a better understanding of real-world scheduling challenges.

Key Components and Features:

Backtracking Algorithm:

Implement a Backtracking-based job scheduling algorithm to explore feasible solutions.

Develop strategies for efficiently searching the solution space, considering constraints and dependencies.

Genetic Algorithm:

Design a Genetic Algorithm for job scheduling, emphasizing the evolution of solutions over multiple generations.

Implement genetic operators such as crossover and mutation to enhance the algorithm's exploration capabilities.

User Interface:

Create an intuitive user interface for users to input job details, dependencies, and constraints.

Visualize the results of both Backtracking and Genetic Algorithm solutions, allowing users to compare outcomes.

Insightful Analytics:

Provide insightful analytics on algorithm performance, showcasing scenarios where one algorithm excels over the other.

Generate reports highlighting the strengths and limitations of each algorithm under various conditions.

Scenario Testing:

Implement a scenario testing module that allows users to simulate different scheduling scenarios.

Enable users to observe how Backtracking and Genetic Algorithms handle various complexities and constraints.

Educational Resources:

Include educational resources within the system to help users understand the underlying principles of Backtracking and Genetic Algorithms.

Provide documentation, tutorials, and visual aids to enhance user knowledge.

Real-world Application:

Integrate real-world scheduling challenges or datasets for users to apply the algorithms to practical scenarios.

Demonstrate the system's adaptability and effectiveness in solving actual scheduling problems.

Potential Technologies:

Programming Languages: Python for algorithm implementation and web-based interface.

Web Framework: Flask or Django for developing the user interface.

Visualization: Matplotlib or Plotly for generating visualizations.

Benefits:

In-depth insights into Backtracking and Genetic Algorithms for Job Scheduling.

Empowered users capable of selecting the most suitable algorithm for specific scenarios.

Practical understanding of algorithmic strengths and limitations in real-world applications.

Similar applications:

1. Calendly

Calendly is a dedicated employee scheduling app that lets you create staff schedules and assign shifts to team members. It automatically checks your team’s availability and assigns client meetings to members based on availability, priority, or equal distribution. You can also use the app’s scheduling dashboard to schedule multiple employees for team events.

Mobile features of interest:

Automated event notifications: Send automated messages to employees via SMS or email in case of any cancellations or changes to schedules or events.

Admin management: Access a central admin dashboard to manage employee scheduling processes or make bulk schedule changes when required. You can also add new admins to oversee staff scheduling for different departments within your business.

Availability settings: Allow your employees to set their availability preferences based on their working hours, breaks, and time zones. This feature ensures employees get scheduled for tasks only during the hours they're available.

2. Acuity Scheduling

Acuity Scheduling is another dedicated employee scheduling app that lets you create, update, and share employee schedules. The app provides a dashboard to add new staff members and create multiple calendars if you have employees in various locations. You can assign a specific staff member to a particular client and notify them by text or email when a service booking gets scheduled.

Mobile features of interest:

Confirmation emails: Send shift confirmation emails to your employees, including the date, time, location, and other relevant information.

Availability customization: Customize the availability of employees for a set duration. You can also share employee availability calendars with clients when they're booking services.

Scheduling limits: Set the hours during which your business can accept client bookings. You can also create other booking restrictions, such as having a maximum number of appointments per week.

3. Jira

Jira is primarily a task and issue tracking application. However, its Scheduler plug-in, which can be purchased from the Atlassian marketplace, allows you to create and track employee schedules, assign tasks to specific employees, and set up recurring schedules. It also lets you prioritize tasks based on deadlines to ensure projects are completed on time.

Mobile features of interest:

Schedule customization: Change created employee schedules by reassigning tasks to other team members in case of disruptions or employee absence.

Service management: Manage service teams and their availability. You can create a work schedule for your service teams and assign work hours to multiple team members.

Planner: Schedule teams and team members for a project, task, or ticket, and find employees by analyzing resource availability, roles, and skills.

4. 7shifts

7shifts is a dedicated employee scheduling app for restaurants. It enables you to create and publish schedules, assign shifts to employees, and notify employees about shift changes via app notifications. It also provides a centralized dashboard to add employees and assign shifts based on their availability.

Mobile features of interest:

Manager log book: Maintain private notes on employee schedules and upcoming maintenance work or events. You also get notifications when your staff needs breaks or may exceed overtime limits, helping prevent noncompliance with labor laws.

Shift pool: Create and manage a pool of employees available at different time slots to find staff to cover last-minute absences, fill open shifts, or accommodate shift swaps.

Availability and time-off requests: Allow employees to submit time-off requests as well as communicate which times in the day or days of the week they can't work because of a conflict.

5. ClockShark

lockShark is a time tracking app with built-in employee scheduling functionality. It offers a drag-and-drop editor to create employee shifts. It lets you notify your team about shift updates or changes through the app or email. Your employees can also use the app to clock in and out, take breaks, and update their availability.

Mobile features of interest:

Employee clock in and clock out: Let your employees clock in and out of their schedules and notify managers and co-workers about their availability during work hours.

Built-in task details: Add task descriptions to work schedules, such as the job details and location. Employees can also add photos and files to their tasks to update their teams.

Employee location tracking: Create a “who’s working now” group to indicate active field workers. This feature allows you to oversee which employees are on-site.

6. QuickBooks Time

QuickBooks Time is another time tracking app with employee scheduling capabilities. It offers a drag-and-drop job scheduler to plan and create employee work schedules. You can create a new employee schedule template or repeat previously created schedules. The app also provides GPS functionality to track employee locations and notifies them of any scheduling changes via text and email notifications.

Mobile features of interest:

Multiple scheduling options: Access various scheduling options such as shift-based scheduling that assigns shifts based on employee availability or job-based scheduling that allocates shifts based on the job type and finds specialists within your team that can handle the tasks.

Remote shift management: Remotely edit and change shift details, such as timing and shift duration. You can also reassign a shift to another worker in case of an emergency.

Scheduling alerts: Send scheduling alerts to your team through in-app and email notifications to remind them about an upcoming shift, job, or scheduled event.

7. Clockify

Clockify is also a time clock app with employee scheduling capabilities. It allows you to create and set up projects, assign projects to team members, visualize project progress through a centralized dashboard, and invite specific members to plan their work schedules. You can also use the project dashboard to manage team capacity by identifying employees who can take up more work or who have too many hours assigned.

Mobile features of interest:

Calendar visualization: Sync your team’s calendar to the project dashboard to track and adjust employee work schedules. You can edit schedules or add more team members to a specific task.

Timesheet management: Let your employees log their weekly activities and schedule work for upcoming weeks. They can also submit timesheets for manager approval and set reminders for due timesheets.

Capacity management: Visualize your team’s workload on the dashboard to track employee capacity, availability, and schedules and distribute tasks equally.

PAPER 1

Page 1

"Genetic Algorithm | SpringerLink Part of the Studies in Computational Intelligence book series (SCI,volume 780)**"**

Genetic Algorithm (GA) is one of the first population-based stochastic algorithm proposed in the history. Similar to other EAs, the main operators of GA are selection, crossover, and mutation. Paper number 1 briefly presents this algorithm and applies it to several case studies to observe its performance. Paper number 1 is actually a chapter out of a book all about genetic algorithm and what inspired people to come up with this method to enhance problem solving. We intend on making Paper 1 a more formal and scholastic research about GA. We however, intend to simplify it even further using an example on Paper 2.

Here we go,

Paper 1 speaks about how GA was inspired from the Darwinian theory of evolutionary, in which the survival of fitter creature and their genes were simulated. We're talking about Evolution of Earth and creatures to becoming what we know of them today. Some made the cut, others gone extinct, GA is a population-based algorithm. Every solution corresponds to a chromosome and each parameter represents a gene. GA evaluates the fitness of each individual in the population using a fitness (objective) function. This is where this becomes "survival of the fittest". For improving poor solutions, the best solutions are chosen randomly with a selection (e.g. roulette wheel) mechanism. This operator is more likely to choose the best solutions since the probability is proportional to the fitness (objective value). What increases local optima avoidance is the probability of choosing poor solutions as well. This means that if good solutions be trapped in a local solution, they can be pulled out with other solutions. However, a lot of us found this concept confusing, so let me clarify:

- A population of potential solutions (individuals) is created. These individuals often have random characteristics to ensure a diverse starting point. This is why we mentioned above that the "options are picked at random".

- Each individual's fitness is evaluated using an objective function specific to the problem being solved. The objective function determines how well each solution performs.

- Individuals are selected for reproduction based on their fitness. Those with higher fitness have a greater chance of being chosen, mimicking the principle of natural selection. Let's say that who ever reaches the end line first wins.

- Pairs of selected individuals exchange genetic information to create new solutions. This process, known as crossover, combines traits from both parents.
- Occasionally, random changes (mutations) are introduced in the new solutions to add diversity and prevent the algorithm from getting stuck in local optima.
- The new solutions replace the old ones, and the cycle repeats for a new generation.

The GA algorithm is stochastic, so one might ask how reliable it is. What makes this algorithm reliable and able to estimate the global optimum for a given problem is the process of maintaining the best solutions in each generation and using them to improve other solutions. As such, the entire population becomes better generation by generation. The crossover between individuals results in exploiting the 'area' between the given two parent solutions. This algorithm also benefits from mutation. This operator randomly changes the genes in the chromosomes, which maintains the diversity of the individuals in the population and increases the exploratory behavior of GA. Similar to the nature, the mutation operator might result is a substantially better solution and lead other solutions towards the global optimum. A main target if we get to call it that (global target).

1. We start with what is commonly known as "initial population", which basically is what we meant when we mentioned the "random solutions". As they are called, the initial population represents our first set of random solution by which we try to find the solutions that work best with us.
2. Every solution then has what I like to call a "manual" where every solutions strength and weaknesses are evaluated to the "fitness function". After the Evaluation we start with what's called "the crossover"
3. The crossover is where the best traits of every solution is then passed to the next generation of solution. The loop keeps going on until we have and Idea of what the best solution to a problem could be. Generations of solutions keep passing on good traits to the generation that follows, until we find what we so call as the best solution.

Finally if this paper's summarization was too formal and hard to understand, paper 2 will be less former using and example intended to make things easier.

PAPER 2

Genetic Algorithm Tom V. Mathew Assistant Professor, Department of Civil Engineering, Indian Institute of Technology Bombay, Mumbai-400076.

Genetic Algorithms: Nature-Inspired Problem Solvers

Genetic Algorithms (Gas) are like digital detectives solving puzzles by mimicking how nature evolves and adapts. Paper number 2 speaks about Genetic Algorithm in General and how it relates with nature in the aspects of natural selection not only inspiring but also being the soul core of the development of this algorithm. Imagine a computer trying different solutions to a problem, learning from the best ones, and mixing them up for even better results.

At the start, Gas create a diverse group of potential solutions, kind of like a team of problem-solving members. Each member has a unique approach. As they tackle the problem, their performance is measured using a special process called the "fitness function." This function tells us which members are doing well.

Gas borrow ideas from how living things evolve. Just as living beings have DNA, our members have something similar called chromosomes. These chromosomes carry the "genetic" information about each solution, determining what works and what doesn't. It's like a manual to their strengths an weaknesses in the aspect of dealing with the problem presented.

Now comes the part where the members of the team start sharing and comparing ideas benefitting from each other's strength and avoiding the weaknesses. They then create a new set (generation) of members that in turn are stronger and more efficient.

Natural selection, where the fittest survive, is a big deal in Gas too. The members that perform better get more chances to "produce another generation" and pass on their good traits. Over time, the team gets smarter and more effective.

To use Gas for specific problems, we need to turn our real-world challenges into a language the computer understands. Think of it like translating our problems into a secret code. Gas work their magic by decoding and recoding this secret language until they crack the problem.

In a nutshell, Gas are like a dynamic team of problem-solvers inspired by nature's way of evolving. They start with a diverse group, learn from the best, mix things up, and keep improving. It's a clever and efficient approach to tackling a wide range of real-world puzzles.

Paper 2 was 15 pages long detailed explanation of the genetic algorithm, how it started, inspired, and implemented. Although it had too much details ,some of which I personally did not understand, I summarized everything I understood in a simpler and more fun way according to my own understanding.

Our project involves the use of another algorithm, which is the backtracking algorithm that we will be discussing in the next page.

PAPER 3

Foundations of Artificial Intelligence
Volume 2, 2006, Pages 85-134

Chapter 4 - Backtracking Search Algorithms

For our paper 3, we decided it would be a chapter of another book.

This chapter explores that there are three main algorithmic techniques for solving constraint satisfaction problems: (1) backtracking search, (2) local search, and (3) dynamic programming. However, our main concern is only about the backtracking algorithm limited by our project's objective.

Backtracking: A Thoughtful Exploration

Imagine you're faced with a challenging puzzle, and you want to find the perfect solution. Backtracking is akin to taking careful steps, making choices, and backtracking if an error is detected. It meticulously explores different possibilities, ensuring that all the rules are followed. If a wrong turn is taken, it gracefully retraces its steps, trying alternative paths until it uncovers a solution. All "nodes" are discovered and tested until the best solution is found.

So in contrast with the GA discussed in the previous 2 papers, the backtracking algorithm tends to try out or "discover" all solutions until the best solution among all solutions is found or until the algorithm reaches its goal.

PAPER 4

Journal of Parallel and Distributed Computing
Volume 70, Issue 1, January 2010, Pages 13-22

Genetic algorithms for task scheduling problem

For paper 4, we decided to get deep into our project. We decided to talk about GA and how it could actually help with our Job scheduling task.

The paper we chose to represent paper number 4 talks about how several genetic algorithms have been developed to solve “the scheduling” problem. A common feature in most of them has been the use of chromosomal representation for a schedule which is basically the main point surrounded by the GA algorithm. However, these algorithms are monolithic (which by the way means “large, powerful, indivisible, and slow to change”), as they attempt to scan the entire solution space without considering how to reduce the complexity of the optimization process. In this paper, two genetic algorithms have been developed and implemented but as we mentioned, we are only discussing a general view of the paper that are concerned with our course and project. However we are going to briefly mention both algorithms discussed in this paper.

This paper discusses how the developed algorithms are genetic algorithms with some heuristic principles (Heuristic principles are general guidelines for evaluating the usability of a product or system.) that have been added to improve the performance. According to the first developed genetic algorithm, two fitness functions have been applied one after the other. The first fitness function is concerned with minimizing the total execution time (schedule length), and the second one is concerned with the load balance satisfaction. The second developed genetic algorithm is based on a task duplication technique to overcome the communication overhead. The proposed algorithms have been implemented and evaluated using benchmarks. According to the evolved results, it has been found that the algorithms always outperform the traditional algorithms.

Now we will just briefly go over the two algorithms mentioned above:

1. Critical Path Genetic Algorithm (CPGA): CPGA's main idea it spots the most important tasks, makes a genius plan to get them done quickly, and ensures the entire workload is handled efficiently. All of this makes your computer tasks run smoothly and fast.
2. Task Duplication Genetic Algorithm (TDGA): TDGA is like the teamwork expert of scheduling algorithms. It duplicates tasks strategically, fills in the waiting gaps, and ensures your computer tasks are not just fast but also work together like a well-coordinated team.

And this was just a brief explanation of the 2 algorithms mentioned on paper number 4 and not to go into details as it is not a priority of our project.

Paper 5

Artificial Intelligence
Volume 76, Issues 1–2, July 1995, Pages 455-480

Backtracking techniques for the job shop scheduling constraint satisfaction problem

For paper 5, we've searched for a paper that relates directly with our project and how we used backtracking algorithm to solve our job scheduling problem, but unfortunately we weren't as lucky as we were with the GA as BT doesn't have as much materials the former. However, we decided to come as close as we could get and pick the parts that relates with our project!

In this study, we explore a specific type of job shop scheduling problem where certain operations must be scheduled within fixed time windows, making it a challenging Constraint Satisfaction Problem (CSP). The conventional approach to solving such problems involves using a depth-first backtrack search. In our previous work, we concentrated on enhancing the efficiency of this search by developing techniques that enforce consistency and heuristics for ordering variables and values. This paper takes a step further by incorporating new look-back strategies that assist the search process in recovering from dead-end search states, which are partial solutions that cannot be completed without violating constraints.

We introduce three intelligent backtracking schemes:

Dynamic Consistency Enforcement:

- Identifies critical subproblems dynamically and decides how far to backtrack by selectively enforcing higher levels of consistency among variables involved in these critical subproblems.

Learning Ordering From Failure:

•Adjusts the order in which variables are instantiated based on past conflicts, learning from failures to improve the decision-making

Reduced Complexity:

They further diminish the average complexity of the backtrack search procedure.

Enhanced Problem Solving:

Our system can efficiently solve problems that were otherwise impractical due to high computational costs.

Effectiveness in Job Shop Scheduling:

Compared to other look-back schemes found in existing literature, these schemes prove to be more effective in solving job shop scheduling problems.
In summary, our study introduces intelligent backtracking strategies that not only improve the efficiency of the search process but also enable the solution of previously challenging problems in job shop scheduling. These approaches outperform other existing look-back schemes, showcasing their effectiveness in addressing complex scheduling scenarios.

And with that we conclude our 5 papers review.

5. Details of the Algorithm(s)/Approach(es) Used and the Results of the Experiments

Backtracking Algorithm:

The Backtracking Algorithm is employed to find an optimal schedule for job instances. Key components of the algorithm include:

is_valid_schedule(schedule): Determines the validity of a given schedule by checking resource capacity and job dependencies.

backtrack(schedule, remaining_jobs): Recursively explores all possible schedules, updating the best schedule when a valid one is found.

solve(): Initiates the backtracking process, and upon completion, displays the optimal schedule or notifies if none is found.

Results: The Backtracking Algorithm tends to find solutions quickly but may not always produce optimal schedules. It is particularly efficient when a valid schedule is found early in the exploration.

Genetic Algorithm:

The Genetic Algorithm aims to discover optimal schedules through evolutionary principles. Key features include:

initialize_population(): Creates an initial population of schedules by randomly assigning jobs to resources.

fitness(schedule): Evaluates the fitness of a schedule based on makespan, penalizing invalid schedules.

evolve(): Iteratively applies selection, crossover, and mutation operations to the population to improve schedules over generations.

Results: The Genetic Algorithm excels in finding optimal schedules, often outperforming the Backtracking Algorithm. However, it might require more time to converge, making it slower in some cases.

Experiment Results:

A comparative study was conducted on random instances:

Backtracking vs. Genetic Algorithm:

Optimal Schedule: Genetic Algorithm frequently outperformed the Backtracking Algorithm in producing optimal schedules.

Execution Time: Backtracking Algorithm generally completed faster due to its nature of exploring possibilities exhaustively.

Note: The trade-off between optimality and speed should be considered based on specific application requirements.

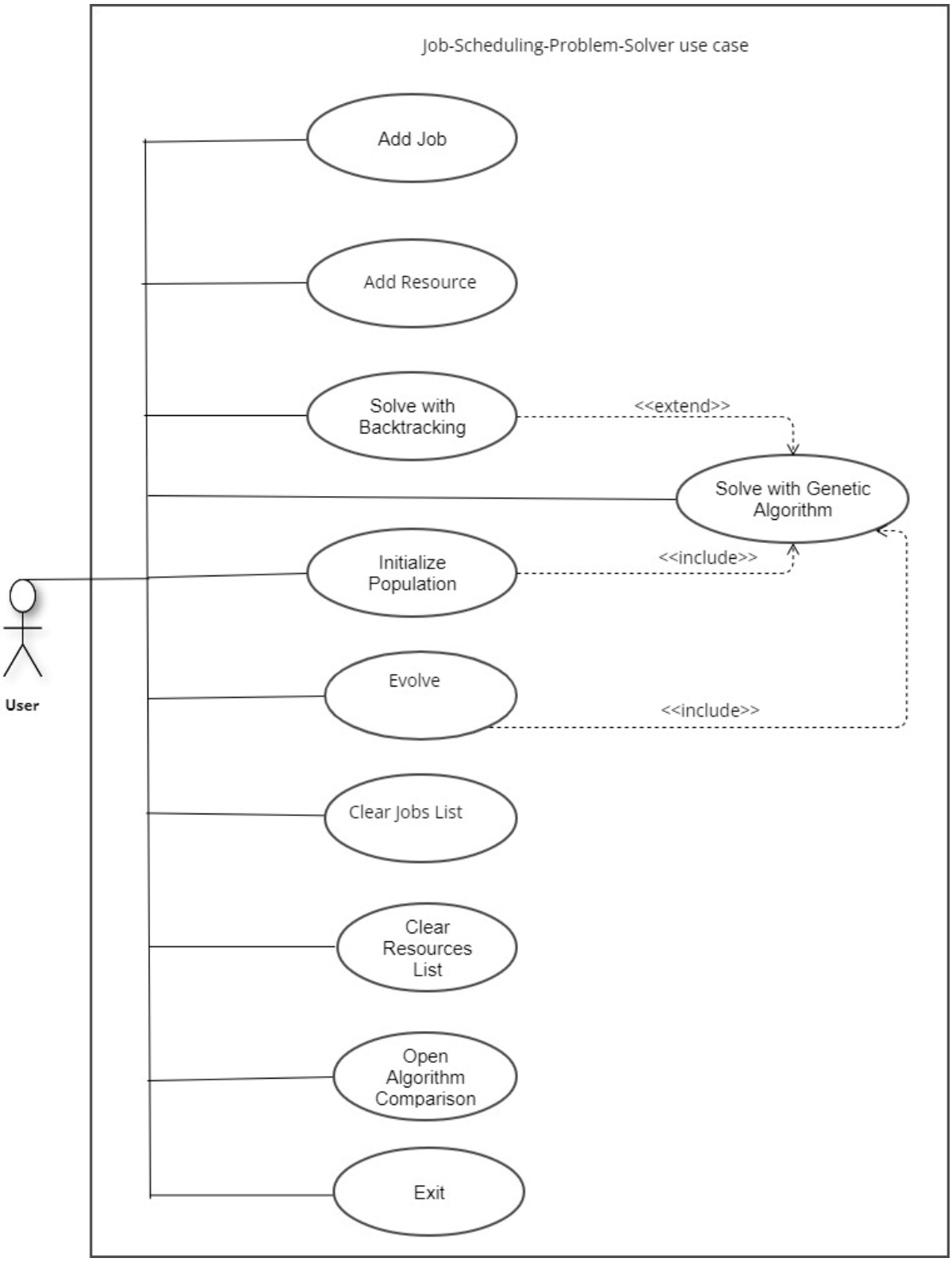
6. Development Platform and Tools Used

Programming Language: Python

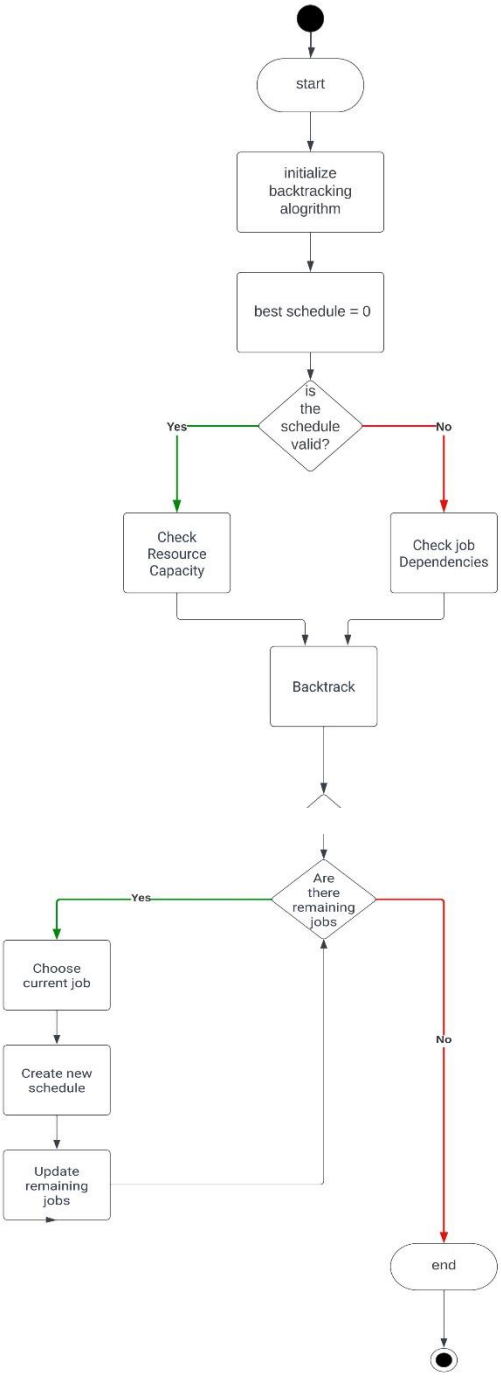
GUI Library: Tkinter

Version Control: Git

Diagrams:


Job-Scheduling-Problem-Solver use case
Add Job
Add Resource
Solve with Backtracking
<<extend>>
Solve with Genetic Algorithm
Initialize Population
<<include>>
Evolve
<<include>>
Clear Jobs List
Clear Resources List
Open Algorithm Comparison
Exit
User

Backtrackig Algorithm



Genetic Algorithm