CS5800: Final Project Report

Optimizing the job application process with priority based scheduling

Aswin Chander Aravind Kumar, Indhuja Muthu Kumar, Keerthana Velilani

Youtube Video Link:

https://youtu.be/g6TqiFXEpfI

Introduction

In the fast-paced realm of job hunting, where opportunities are diverse and timelines are stringent, the quest to optimize the job application process has never been more crucial. Our project is a dedicated effort to bring efficiency and strategic organization to this intricate journey by introducing a priority-based scheduling approach.

In this pursuit, our project acknowledges the fast-paced nature of the job market, recognizing the significance of adaptability and precision. By embracing a priority-based scheduling approach, we aim to empower individuals in their pursuit of professional opportunities, offering a solution that not only categorizes and organizes job prospects but does so with an understanding of deadlines and individual preferences.
`

Aswin:
Embarking on the co-op and job search journey as a graduate student in robotics brought about its fair share of challenges. Balancing my coursework, assignments, and the job hunt turned into a bit of a hassle. I found myself struggling to connect the dots between job descriptions and my resume, analyzing the skills needed for different roles, and, of course, managing the scheduling of interviews.

It became clear that a solution tailored to our unique needs as students was essential. A system that could not only categorize job opportunities but also prioritize them based on upcoming deadlines would be a huge breakthrough. This wasn't just about simplifying the process; it was about easing the constant struggle between academic responsibilities and the pursuit of professional opportunities in robotics.

Indhuja:
Navigating my job search as an international master's student in Data Science presented challenges, particularly when an exceptional opportunity arose, demanding expertise in cutting-edge machine learning and advanced data structures. The tight application window, coinciding with an industry summit on artificial intelligence, led to a

hastily prepared application that later revealed its shortcomings. This experience underscored the necessity for a tailored scheduling algorithm, prioritizing the optimization of the job application process. This algorithm aims to strategically plan application content, alert users to unconventional deadlines, and ensure applications align precisely with the technical requirements of each opportunity.

Keerthana:
The topic of tracking job application deadlines and scheduling holds significant personal relevance for me as a graduate student. Juggling job search, coursework, and necessary commitments leaves me consistently short on time. Navigating the uncertainties inherent in the job search process, particularly as an international student, highlights the importance of optimizing our approach. By developing a system that prioritizes applications based on approaching deadlines using approaches like Priority-based scheduling, this project aims to empower students like myself with greater control over the application process, allowing for efficient time management and increased strategic focus in a competitive job market.

Problem Statement

The objective of the project is to optimize the efficiency of handling and prioritizing various job applications submitted by individuals. The scheduling of these jobs will be based on factors such as deadlines and individual preferences, skill match, experience, and much more. The prioritized jobs can then be utilized as input for scheduling interviews to eliminate conflicts and streamline the overall application process. This project focuses on evaluating the technical proficiency of the algorithm in optimizing the scheduling process.

In essence, the project aims to address the crucial question: "How can job application search be efficiently managed, considering job type, preferences, and deadlines, using a relevant scheduling algorithm and how the interview planning can be more streamlined with graphical visualizations?"

Implementation

Priority based scheduling

Priority-based scheduling is an algorithm used in computer systems where tasks or processes are assigned priority values, and the scheduler selects the task with the highest priority for execution. This approach enables the system to prioritize tasks based on factors like importance, deadlines, or resource requirements. The scheduling can be preemptive, allowing higher-priority tasks to interrupt lower-priority ones, or non-preemptive, where tasks run until completion or voluntarily release the CPU. Priority-based scheduling provides a flexible way to manage system resources but requires careful handling to avoid issues such as priority inversion.

Description of the methods

The project employs below methods to calculate application priorities based on factors such as user preferences, skills match, company reputation, education level, location preference, and diversity.

The **calculate_priority** function assigns weights to these factors and returns a calculated priority for each application.

The **prioritize_applications** method takes a list of job applications and user skills, sorting the applications based on their calculated priorities. The result is a prioritized list of job applications.

The **schedule_applications** method schedules the prioritized applications by considering the next available date and a user-defined threshold for the number of applications per day. The result includes two lists: result, containing the scheduled applications, and missed_applications, containing those that couldn't be scheduled.

To visualize the scheduling outcomes, the script includes two plotting methods. The **plot_pie_chart** function generates a pie chart displaying the distribution of scheduled applications by job type, while the **plot_timeline** function creates a timeline chart illustrating the scheduled dates for each application. These visualizations offer insights into the distribution and timeline of scheduled job applications.

Input

The input applications have been created and considered to mirror real-world scenarios and diverse use cases. It incorporates crucial information such as company name, company reputation, deadline, diversity factor, education level, job_id, job_type, location preference, preference, and required skills for a set of 20 applications. Each criterion is thoughtfully scored, accurately capturing the nuanced needs and preferences of applicants. This thorough set of inputs provides the necessary foundation for the weighted scoring algorithm to calculate priorities with precision, ensuring a thorough evaluation of each job application.

Implementation

Firstly, the application priorities are determined using a weighted scoring algorithm. This algorithm takes into account various factors crucial for application evaluation. These factors include deadlines, candidate preferences, specific skills required for the job and many more. By assigning weights to each of these elements, we can calculate a comprehensive priority score for each application.

Once the priority scores are determined, the system sorts the applications based on these calculated priorities. This ensures that applications are not only evaluated thoroughly but also ranked according to their relevance and urgency.

The prioritize_applications function sorts job applications based on priorities, which is a form of divide-and-conquer strategy to break down the problem into smaller, manageable tasks and is referenced in Module 4: Divide-and-conquer 2 of the course.

Next, the system employs a scheduling algorithm. This algorithm is designed to handle deadlines and conflicts seamlessly. It takes into consideration the time sensitivity of each application, ensuring that deadlines are met while avoiding any scheduling conflicts that may arise. Referring Module 11.2: Survey design and Airline Scheduling gave us insights on scheduling approaches although we used our own implementation for the project.

The algorithm is designed with efficiency in mind. The threshold, set at 2 applications per day, is a crucial parameter. This means the system will schedule up to 2 applications each day based on their priority scores and deadlines. If there are more applications that can be accommodated within the deadline, the algorithm passes on the excess to the next day.

Output
The timeline plot displays scheduled applications over time, giving a visual representation of the workload. The axis displays the dates with the line graph depicting the scheduled job IDs for each respective date. The higher-priority job IDs are presented first on the line, followed by those with lower prioritization.

Additionally, a pie chart breaks down applications scheduled by job type, providing insights into the distribution of tasks across different roles.

Analysis

This project addresses the challenge of efficiently prioritizing job applications and scheduling interviews. The core algorithm, encapsulated in the calculate_priority function, employs a weighted scoring system considering factors like application deadline, applicant preferences, skills match, experience, company reputation, and more. The prioritize_applications function utilizes this prioritization to sort the applications, creating a prioritized list for further processing. The interview scheduling system, implemented in schedule_interviews, leverages the prioritized list to assign interview dates, avoiding conflicts and ensuring optimal candidate selection. The visual representation in plot_scheduled_interviews includes a line plot illustrating the distribution of interviews over time and a pie chart depicting skill matches for each job type. This holistic approach integrates diverse criteria, ensuring a comprehensive evaluation of applications. The project emphasizes algorithmic efficiency and clarity, aligning with the principles taught in the course.
The project demonstrates an overall efficient algorithmic design with a time complexity of O(NlogN) for the prioritization and interview scheduling process, and a space complexity of O(N) due to the storage of sorted applications and scheduled interviews.

Results

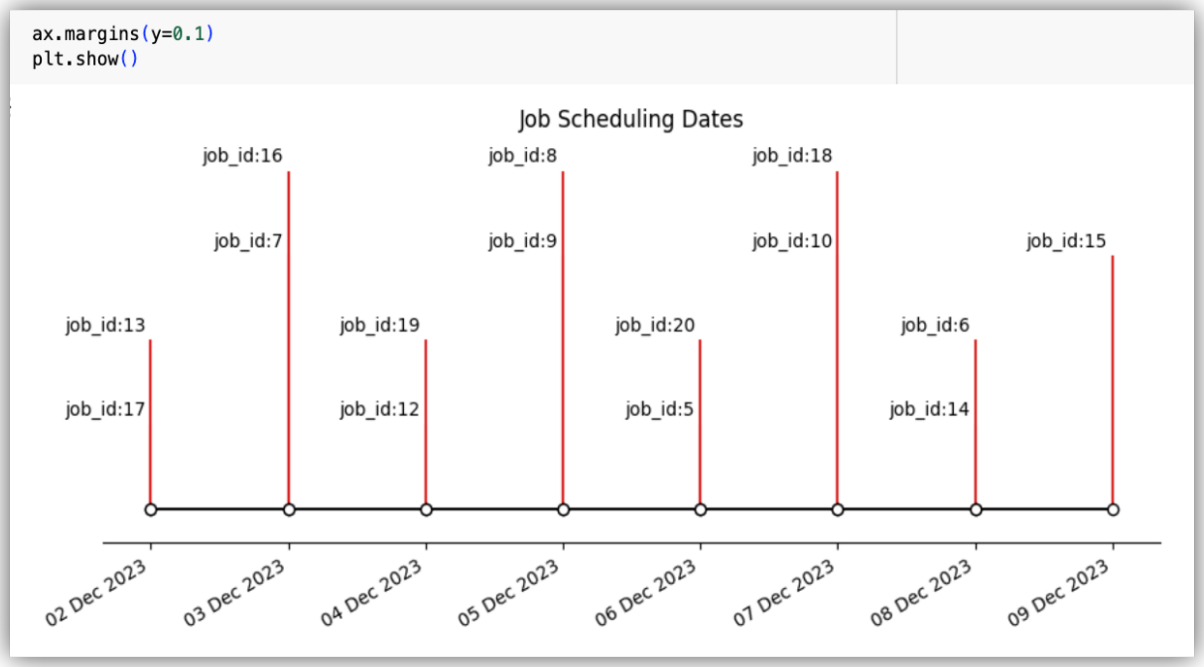
Scheduled applications

```
Final schedule:
[{'company_name': 'Maxar Technologies',
  'days_for_deadline': 1,
  'job_id': 17,
  'priority': 16.423333333333332,
  'suggested_date': datetime.datetime(2023, 12, 2, 16, 31, 52, 692001)},
 {'company_name': 'Costco',
  'days_for_deadline': 23,
  'job_id': 13,
  'priority': 16.232424242424244,
  'suggested_date': datetime.datetime(2023, 12, 2, 16, 31, 52, 692001)},
 {'company_name': 'Boeing',
  'days_for_deadline': 2,
  'job_id': 7,
  'priority': 15.923333333333334,
  'suggested_date': datetime.datetime(2023, 12, 3, 16, 31, 52, 692001)},
 {'company_name': 'Pendulum',
  'days_for_deadline': 5,
  'job_id': 16,
  'priority': 15.773333333333335,
  'suggested_date': datetime.datetime(2023, 12, 3, 16, 31, 52, 692001)},
 {'company_name': 'Amazon',
  'days_for_deadline': 16,
  'job_id': 12,
  'priority': 15.736666666666668,
  'suggested_date': datetime.datetime(2023, 12, 4, 16, 31, 52, 692001)},
 {'company_name': 'Tik Tok',
  'days_for_deadline': 11,
  'job_id': 19,
  'priority': 13.653333333333332,
  'suggested_date': datetime.datetime(2023, 12, 4, 16, 31, 52, 692001)},
 {'company_name': 'Twill',
  'days_for_deadline': 13,
  'job_id': 9,
  'priority': 13.65,
  'suggested_date': datetime.datetime(2023, 12, 5, 16, 31, 52, 692001)},
 {'company_name': 'Meta',
  'days_for_deadline': 18,
  'job_id': 8,
  'priority': 13.645098039215686,
  'suggested_date': datetime.datetime(2023, 12, 5, 16, 31, 52, 692001)},
 {'company_name': 'Rekovar Inc',
  'days_for_deadline': 8,
  'job_id': 5,
  'priority': 13.075238095238097,
  'suggested_date': datetime.datetime(2023, 12, 6, 16, 31, 52, 692001)},
 {'company_name': 'Netflix',
  'days_for_deadline': 8,
  'job_id': 20,
  'priority': 13.075238095238097,
  'suggested_date': datetime.datetime(2023, 12, 6, 16, 31, 52, 692001)},
```

```
  'suggested_date': datetime.datetime(2023, 12, 6, 16, 31, 52, 692001)},
 {'company_name': 'Microsoft',
  'days_for_deadline': 12,
  'job_id': 10,
  'priority': 13.064848484848486,
  'suggested_date': datetime.datetime(2023, 12, 7, 16, 31, 52, 692001)},
 {'company_name': 'LinkedIn',
  'days_for_deadline': 14,
  'job_id': 18,
  'priority': 12.382051282051282,
  'suggested_date': datetime.datetime(2023, 12, 7, 16, 31, 52, 692001)},
 {'company_name': 'Intuit',
  'days_for_deadline': 28,
  'job_id': 14,
  'priority': 12.374074074074073,
  'suggested_date': datetime.datetime(2023, 12, 8, 16, 31, 52, 692001)},
 {'company_name': 'MedSpeed',
  'days_for_deadline': 11,
  'job_id': 6,
  'priority': 11.386666666666667,
  'suggested_date': datetime.datetime(2023, 12, 8, 16, 31, 52, 692001)},
 {'company_name': 'Leidos',
  'days_for_deadline': 23,
  'job_id': 15,
  'priority': 9.875757575757575,
  'suggested_date': datetime.datetime(2023, 12, 9, 16, 31, 52, 692001)}]
-------------------------
missed_applications:
[{'company_name': 'WayFair',
  'days_for_deadline': -1,
  'job_id': 1,
  'job_type': 'Software Developer',
  'priority': 16.423333333333332},
 {'company_name': 'Airtable',
  'days_for_deadline': 2,
  'job_id': 2,
  'job_type': 'Data Scientist',
  'priority': 13.746666666666668},
 {'company_name': 'Glass Imaging',
  'days_for_deadline': 4,
  'job_id': 3,
  'job_type': 'Machine Learning Engineer',
  'priority': 12.7},
 {'company_name': 'Cardio Diagnostics Holdings Inc.',
  'days_for_deadline': 0,
  'job_id': 11,
  'job_type': 'Data Engineer',
  'priority': 12.566666666666666},
 {'company_name': 'Salesforce',
  'days_for_deadline': 3,
  'job_id': 4,
  'job_type': 'Data Engineer',
  'priority': 12.466666666666665}]
-------------------------
Scheduled/Total=15/20
```

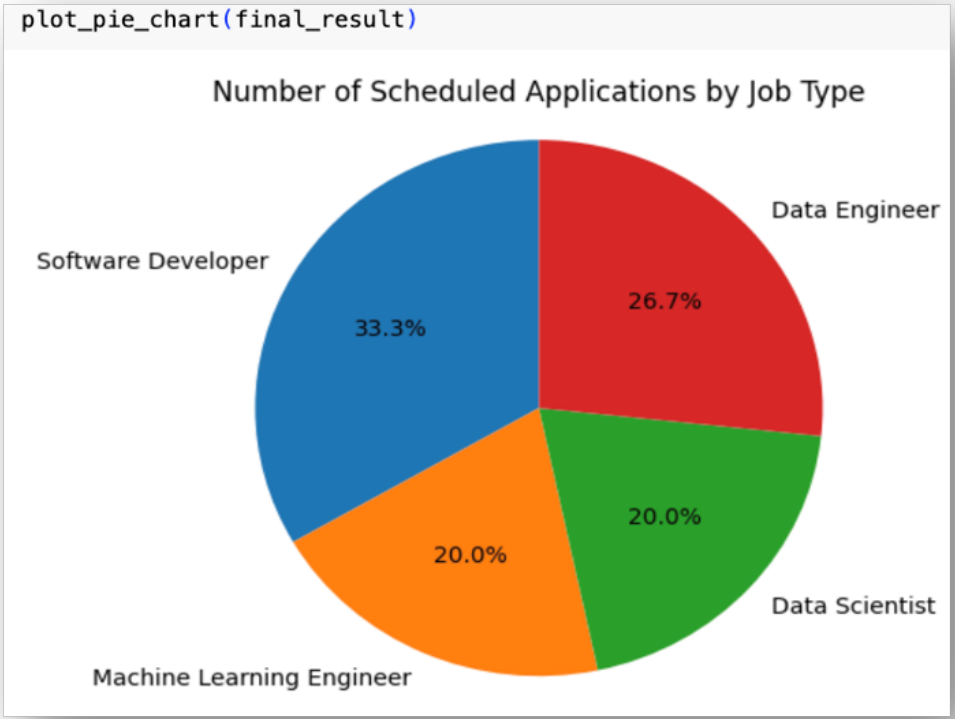
Timeline of scheduled applications

```
ax.margins(y=0.1)
plt.show()
```



Pie chart of scheduled applications by job type

```
plot_pie_chart(final_result)
```



Conclusion

In conclusion, the project successfully addresses the primary objective of optimizing the efficiency of handling job applications and streamlining the interview scheduling process. The implemented algorithm effectively prioritizes applications based on a comprehensive set of criteria, including preferences, skills match, deadlines, and company reputation. The subsequent scheduling of interviews takes into account these priorities, leading to an organized and conflict-free interview calendar.

The analysis reveals that the algorithm performs efficiently, demonstrating a time complexity of O(NlogN) for the prioritization and scheduling processes. The space complexity, at O(N), remains reasonable, given the storage requirements for sorted applications and scheduled interviews. The visualizations further enhance the project's transparency, providing a clear representation of the distribution of interviews over time and the skill matches across different job types.

One limitation is that the algorithm could benefit from further refinement to handle real-world complexities, such as unexpected changes in applicant availability or company-specific constraints.

In the future, the project holds significant promise for growth and improvement. An important step forward involves integrating the job application task scheduling system with common tracking tools, like calendar applications, to provide timely alerts based on priority deadlines. This feature aims to offer applicants organized notifications, facilitating better adherence to deadlines. Moreover, leveraging Natural Language Processing (NLP) algorithms to identify preference keywords would enhance the system's ability to extract valuable insights from applicants' preferences, further refining prioritization. Additionally, the integration of graphical representations, like charts and graphs, would provide a visual depiction of patterns and trends in the job search process. Thus, these pragmatic enhancements, including seamless integration with tracking tools, refined NLP algorithms, and graphical representations, collectively underline the project's commitment to fostering a more efficient and user-friendly job application management experience.

Self Reflection

Aswin

Working on this project has significantly enriched my experience of working with algorithms. It has advanced my abilities in developing complex scheduling algorithms that consider various critical factors. My proficiency in assessing technical capabilities has been refined through evaluation of performance indicators and optimization methods. Additionally, delving into graphical visualizations has expanded my aptitude for creating user-friendly interfaces that are both functionally robust and intuitive. Overall, this project has been an invaluable exercise in algorithmic innovation. Beyond personal development,I feel the expertise I've gained will also be instrumental in enhancing my future endeavors.

Indhuja

Participating in this project has been an enriching learning experience. It has heightened my expertise in crafting intricate scheduling algorithms, taking into account factors such as deadlines and individual preferences. Through the evaluation of technical proficiency, I have honed my skills in performance metrics and optimization strategies. The process of streamlining interview planning has further enhanced my skills in designing efficient algorithms. Exploring graphical visualizations has broadened my understanding of creating interfaces that are both technically sound and user-friendly. In summary, this project has been a valuable journey of algorithmic discovery, significantly enhancing my professional skill set. Importantly, it has not only contributed to my professional growth but has also proven advantageous in my ongoing job search.

Keerthana

This project has been instrumental in deepening my understanding of algorithmic optimization techniques applied to job applications. I honed my skills in optimizing decision-making processes, particularly in weighing factors like skills, preferences, deadlines and scheduling algorithms. Moving forward, the integration of machine learning algorithms stands out as a potential enhancement, enabling adaptive adjustments to criteria weights based on historical application data. This technical exploration has refined my algorithmic proficiency and directly influenced my ongoing job search by instilling a data-driven and strategic approach to the process, offering a practical solution to real-world challenges.

Appendix

Project Code

The **calculate_priority** function assigns weights to these factors and returns a calculated priority for each application.

```python
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from pprint import pprint
import numpy as np

job_priorities = {}

# Calculate priority of each application
def calculate_priority(application, user_skills):

    # Weights for each feature
    preference_weight = 1.0
    skills_weight = 0.8
    reputation_weight = 0.6
    early_submission_bonus = 0.2
    education_weight = 0.5
    location_preference_weight = 0.5
    diversity_weight = 0.3
    skill_match_threshold = 0.6

    days_left = (application['deadline'] - datetime.now()).days

    # User preference values
    user_skills_set = set(user_skills)
    required_skills_set = set(application['required_skills'])
    skills_match = len(user_skills_set.intersection(required_skills_set)) / len(required_skills_set) if required_skills_set else 0
    education_level_values = {'PhD': 3, 'Master': 5, 'Bachelor': 4, 'High School': 2, 'Other': 1}
    education_level_value = education_level_values.get(application['education_level'], 0)
    location_preference_values = {'Urban': 5, 'Suburban': 4, 'Rural': 3}
    location_preference_value = location_preference_values.get(application['location_preference'], 0)

    # Calculate priority using weights and values for each feature
    priority = (
        (preference_weight * application['preference']) +
        (skills_weight * skills_match) +
        (reputation_weight * application['company_reputation']) +
        (early_submission_bonus / max(1, days_left)) +
        (education_weight * education_level_value) +
        (location_preference_weight * location_preference_value) +
        (diversity_weight * application['diversity_factor'])
    )

    job_priorities[application['job_id']] = priority
    return priority

```

The **prioritize_applications** method takes a list of job applications and user skills, sorting the applications based on their calculated priorities.

```python
# Sort prioritized applications
def prioritize_applications(applications, user_skills):
    sorted_applications = sorted(applications, key=lambda app: calculate_priority(app, user_skills), reverse=True)
    return sorted_applications

```

The **schedule_applications** method schedules the prioritized applications by considering the next available date and a user-defined threshold for the number of applications per day

```
# Schedule prioritized applications based on next available date and user threshold for number of applications per day
def schedule_applications(prioritized_applications):
    current_time = datetime.now()
    schedule = []
    jobid_deadline_dictionary = {} # Jobid - deadline dictionary
    calendar = {}
    apps_per_date = {} #Track applications per date
    threshold = 2 # Customizable threshold number of applications per day
    day_counter = current_time
    result = []
    missed_applications = []

    for application in prioritized_applications:
        days_until_deadline = (application['deadline'] - current_time + timedelta(days=1)).days
        jobid_deadline_dictionary[application['job_id']] = days_until_deadline

    for application in prioritized_applications:
        while day_counter in apps_per_date and apps_per_date[day_counter] >= threshold:
            day_counter += timedelta(days=1)
        rem_days = (application['deadline'] - day_counter).days
        if rem_days >= 0:
            calendar[application['job_id']] = day_counter
            result.append({'job_id': application['job_id'], 'days_for_deadline': jobid_deadline_dictionary[application['job_id']],
                       'suggested_date': calendar[application['job_id']], 'priority': job_priorities[application['job_id']],
                       'company_name': application['company_name']})

            apps_per_date[day_counter] = 1 + apps_per_date.get(day_counter, 0)
        else:
            missed_applications.append({'job_id': application['job_id'], 'days_for_deadline': jobid_deadline_dictionary[application['job_id']], 'job_type': application['job_type'],
                                'priority': job_priorities[application['job_id']],'company_name': application['company_name']})

    return result, missed_applications

```

The **plot_pie_chart** function generates a pie chart displaying the distribution of scheduled applications by job type,

```
# Pie chart of scheduled applications by Job Type
def plot_pie_chart(final_result):
    job_types_count = {}
    for application in final_result:
        job_type = job_applications[application['job_id'] - 1]['job_type']
        job_types_count[job_type] = job_types_count.get(job_type, 0) + 1

    labels = job_types_count.keys()
    sizes = job_types_count.values()

    fig, ax = plt.subplots()
    ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
    ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

    plt.title("Number of Scheduled Applications by Job Type")
    plt.show()

```

The **plot_timeline** function creates a timeline chart illustrating the scheduled dates for each application.

```python
# Timeline of scheduled applications
def plot_timeline(final_result):
  dates = []
  names = []
  threshold = 2

  for app in final_result:
      job_id = app['job_id']
      suggested_date = app['suggested_date']
      date_string = suggested_date.strftime('%Y-%m-%d')
      dates.append(date_string)
      names.append('job_id:'+ str(app['job_id']))

  dates = [datetime.strptime(d, "%Y-%m-%d") for d in dates]

  levels = np.tile([1, 2, 3, 4],
                   int(np.ceil(len(dates) / 4)))[:len(dates)]

  fig, ax = plt.subplots(figsize=(8.8, 4), constrained_layout=True)
  ax.set(title="Job Scheduling Dates")

  ax.vlines(dates, 0, levels, color="tab:red")
  ax.plot(dates, np.zeros_like(dates), "-o",
          color="k", markerfacecolor="w")

  for d, l, r in zip(dates, levels, names):
      ax.annotate(r, xy=(d, l),
                  xytext=(-3, np.sign(l) * 3), textcoords="offset points",
                  horizontalalignment="right",
                  verticalalignment="bottom" if l > 0 else "top")

  ax.xaxis.set_major_locator(mdates.DayLocator())
  ax.xaxis.set_major_formatter(mdates.DateFormatter("%d %b %Y"))
  plt.setp(ax.get_xticklabels(), rotation=30, ha="right")

  ax.yaxis.set_visible(False)
  ax.spines[["left", "top", "right"]].set_visible(False)

  ax.margins(y=0.1)
  plt.show()
```

Input (user skills and job applications)

```
if __name__ == "__main__":

    # User skillsets defined here
    user_skills = ['Python', 'SQL', 'Communication', 'Machine Learning']

    # Input job applications to be prioritized and scheduled
    job_applications = [
    {'job_id': 1, 'job_type': 'Software Developer', 'deadline': datetime(2023, 12, 1), 'preference': 10, 'required_skills': ['Python', 'Java', 'SQL'], 'company_reputation': 0.8, 'education_level': 'Master', 'location_preference': 'Urban', 'diversity_factor': 0.7, 'company_name': 'WayFair'},

    {'job_id': 2, 'job_type': 'Data Scientist', 'deadline': datetime(2023, 12, 4), 'preference': 9 , 'required_skills': ['Python', 'Data Visualization', 'Mathematics'], 'company_reputation': 0.9, 'education_level': 'PhD','location_preference': 'Suburban', 'diversity_factor': 0.8, 'company_name': 'Airtable'},

    {'job_id': 3, 'job_type': 'Machine Learning Engineer', 'deadline': datetime(2023, 12, 6), 'preference': 8, 'required_skills': ['Machine Learning', 'Algorithm', 'Python'], 'company_reputation': 0.7, 'education_level': 'Bachelor','location_preference': 'Rural', 'diversity_factor': 0.6, 'company_name': 'Glass Imaging'},

    {'job_id': 4, 'job_type': 'Data Engineer', 'deadline': datetime(2023, 12, 5), 'preference': 7, 'required_skills': ['Data Structures', 'Big Data', 'SQL'], 'company_reputation': 0.7, 'education_level': 'Bachelor','location_preference': 'Urban', 'diversity_factor': 0.6, 'company_name': 'Salesforce'},

    {'job_id': 5, 'job_type': 'Data Scientist', 'deadline': datetime(2023, 12, 10), 'preference': 9 , 'required_skills': ['Python', 'Data Visualization', 'Mathematics'],'company_reputation': 0.9, 'education_level': 'PhD', 'location_preference': 'Rural', 'diversity_factor': 0.8, 'company_name': 'Rekovar Inc'},

    {'job_id': 6, 'job_type': 'Data Engineer', 'deadline': datetime(2023, 12, 13), 'preference': 7, 'required_skills': ['Data Structures', 'Big Data', 'SQL'], 'company_reputation': 0.7, 'education_level': 'Bachelor', 'location_preference': 'Rural', 'diversity_factor': 0.6, 'company_name': 'MedSpeed'},

    {'job_id': 7, 'job_type': 'Software Developer', 'deadline': datetime(2023, 12, 4), 'preference': 10, 'required_skills': ['Python', 'Java', 'SQL'], 'company_reputation': 0.8, 'education_level': 'Master', 'location_preference': 'Suburban', 'diversity_factor': 0.7, 'company_name': 'Boeing'},

    {'job_id': 8, 'job_type': 'Machine Learning Engineer', 'deadline': datetime(2023, 12, 20), 'preference': 8, 'required_skills': ['Machine Learning', 'Algorithm', 'Python'],'company_reputation': 0.7, 'education_level': 'Bachelor','location_preference': 'Urban', 'diversity_factor': 0.6, 'company_name': 'Meta'},

    {'job_id': 9, 'job_type': 'Machine Learning Engineer', 'deadline': datetime(2023, 12, 15), 'preference': 8, 'required_skills': ['Machine Learning', 'Algorithm', 'Python'],'company_reputation': 0.7, 'education_level': 'Bachelor', 'location_preference': 'Urban', 'diversity_factor': 0.6, 'company_name': 'Twill'},

    {'job_id': 10, 'job_type': 'Data Scientist', 'deadline': datetime(2023, 12, 14), 'preference': 9 , 'required_skills': ['Python', 'Data Visualization', 'Mathematics'], 'company_reputation': 0.9, 'education_level': 'PhD', 'location_preference': 'Rural', 'diversity_factor': 0.8, 'company_name': 'Microsoft'},

    {'job_id': 11, 'job_type': 'Data Engineer', 'deadline': datetime(2023, 12, 2), 'preference': 7, 'required_skills': ['Data Structures', 'Big Data', 'SQL'], 'company_reputation': 0.7, 'education_level': 'Bachelor', 'location_preference': 'Urban', 'diversity_factor': 0.6, 'company_name': 'Cardio Diagnostics Holdings Inc.

    {'job_id': 12, 'job_type': 'Software Developer', 'deadline': datetime(2023, 12, 10), 'preference': 10, 'required_skills': ['Python', 'Java', 'SQL'],'company_reputation': 0.8, 'education_level': 'Master','location_preference': 'Suburban', 'diversity_factor': 0.7, 'company_name': 'Amazon'},

    {'job_id': 13, 'job_type': 'Software Developer', 'deadline': datetime(2023, 12, 25), 'preference': 10, 'required_skills': ['Python', 'Java', 'SQL'],'company_reputation': 0.8, 'education_level': 'Master','location_preference': 'Urban', 'diversity_factor': 0.7, 'company_name': 'Costco'},

    {'job_id': 14, 'job_type': 'Data Engineer', 'deadline': datetime(2023, 12, 30), 'preference': 7, 'required_skills': ['Data Structures', 'Big Data', 'SQL'], 'company_reputation': 0.7, 'education_level': 'Bachelor','location_preference': 'Urban', 'diversity_factor': 0.6, 'company_name': 'Intuit'},

    {'job_id': 15, 'job_type': 'Data Engineer', 'deadline': datetime(2023, 12, 25), 'preference': 7, 'required_skills': ['Data Structures', 'Big Data', 'SQL'], 'company_reputation': 0.7, 'education_level': 'Bachelor','location_preference': 'suburban', 'diversity_factor': 0.6, 'company_name': 'Leidos'},

    {'job_id': 16, 'job_type': 'Software Developer', 'deadline': datetime(2023, 12, 7), 'preference': 10, 'required_skills': ['Python', 'Java', 'SQL'],'company_reputation': 0.8, 'education_level': 'Master','location_preference': 'Suburban', 'diversity_factor': 0.7, 'company_name': 'Pendulum'},

    {'job_id': 17, 'job_type': 'Software Developer', 'deadline': datetime(2023, 12, 3), 'preference': 10, 'required_skills': ['Python', 'Java', 'SQL'],'company_reputation': 0.8, 'education_level': 'Master','location_preference': 'Urban', 'diversity_factor': 0.7, 'company_name': 'Maxar Technologies'},

    {'job_id': 18, 'job_type': 'Data Engineer', 'deadline': datetime(2023, 12, 16), 'preference': 7, 'required_skills': ['Data Structures', 'Big Data', 'SQL'], 'company_reputation': 0.7, 'education_level': 'Bachelor','location_preference': 'Urban', 'diversity_factor': 0.6, 'company_name': 'LinkedIn'},

    {'job_id': 19, 'job_type': 'Machine Learning Engineer', 'deadline': datetime(2023, 12, 13), 'preference': 8, 'required_skills': ['Machine Learning', 'Algorithm', 'Python'],'company_reputation': 0.7, 'education_level': 'Bachelor','location_preference': 'Urban', 'diversity_factor': 0.6, 'company_name': 'Tik Tok'},

    {'job_id': 20, 'job_type': 'Data Scientist', 'deadline': datetime(2023, 12, 10), 'preference': 9 , 'required_skills': ['Python', 'Data Visualization', 'Mathematics'], 'company_reputation': 0.9, 'education_level': 'PhD','location_preference': 'Rural', 'diversity_factor': 0.8, 'company_name': 'Netflix'}

]
```

Driver code

```
prioritized_applications = prioritize_applications(job_applications, user_skills)
final_result, missed_applications = schedule_applications(prioritized_applications)
print('Final schedule:')
pprint(final_result)
print('--------------------------')
print('missed_applications:')
pprint(missed_applications)
print('--------------------------')
print('Scheduled/Total='+str(len(final_result))+'/'+str(len(job_applications)))

plot_timeline(final_result)
plot_pie_chart(final_result)
```