INTRODUCTION

HKUST offers a rich variety of courses. However, with such a large selection, it becomes difficult to decide which courses to take and when. Students often find that the courses that they wish to take are full and poor scheduling results in students having days with an inconveniently large number of classes. Furthermore, students that choose to take additional or less credits in a semester have difficulty planning which courses fulfill their requirements (as different courses carry different amounts of credits). Finally, students must take the courses that fulfill their graduation requirements, which includes major requirements and university common core requirements. With so many different requirements, students often have trouble scheduling classes. While the Student Information System does provide a decent solution to this problem, it does not provide an efficient way to plan classes and avoid time clashes.

This project aims to facilitate the course selection needs of the student. It is, essentially, a class-scheduling program that assigns a “utility” to each course that is determined by factors set by the student.

The program assigns a utility to each course and it is influenced by factors set by the student, namely:

1) Course priority: This entails three levels, ranging from “must-take” to “take if no other option”.
2) Timing: check availability of slots and add points to the utility if a course has open slots at desirable times, i.e. Friday afternoon classes, avoid 9 a.m morning classes.
3) Credit number: Number of credits you wish to take for the semester

The user inputs all the courses he wishes to take including the lecture times, and assigns the priorities. The program then attempts to analyse time slot violations and it sorts the course selections until it finds a combination of courses and course timings that do not clash. The program ideally returns an array of courses best suited to the needs and priorities mentioned by the student.

ALGORITHM

POSSIBLE CHALLENGES

The team considered some possible challenges that were liable to cause difficulty, prior to the design of the algorithm:

After considering the complexity of the problem, we realized that the size of input N was quite small. Upon calculation, for the average user, the number of unique assignments is less than 15 choices, 5 times 10, 30000 unique assignments, which would be cut down even further by a CSP that checks for time clashes. In addition to this, the schedule planning is a not a particularly complex problem. There are almost no cases where assigning a high priority course will cause the remaining courses to be unassignable, which was an issue that the team was concerned would pose a problem. This means that the majority of searching is done in the lower parts of the tree, for the final few assignments.

IMPLEMENTATION

Due to the possible challenges mentioned above, and time constraints, the team made the decision to utilize a greedy search as the primary search algorithm, aided by CSP to conduct time clashing checks.

Other algorithms like linear programming optimization, or a star heuristic were considered and determined to be faster, however when considering the level of complexity and difficulty of the project, greedy search was deemed sufficient. The structure to be searched is not overly complex and does not possess a large number of substructures, thus, the greedy search serves the purpose of the project.

During the search, course objects are generated for each permutation of the course. The algorithm sorts the courses by priority (as defined by the user), then picks the highest priority courses to assign and explore first. If the assignment would cause a time clash or the names are the same (it is not possible to register the same course multiple times), then the algorithm will roll back one. This loop continues until a valid assignment is achieved, which, in the definition of the project is the optimal output (the highest priority courses have all been selected).
The most intricate part of the algorithm occurs in the choice of priority values. The higher priority course will always be selected if there exists a solution that satisfies the CSP. Therefore, tactical choices need to be made when dealing with courses of the same priority.

Due to time constraints several features that the team wished to add could not be implemented. This includes time slot prioritization and avoiding long stretches of dead time.