Optimizing University Course Schedules Using Evolution Strategies

Sindhu Jampani

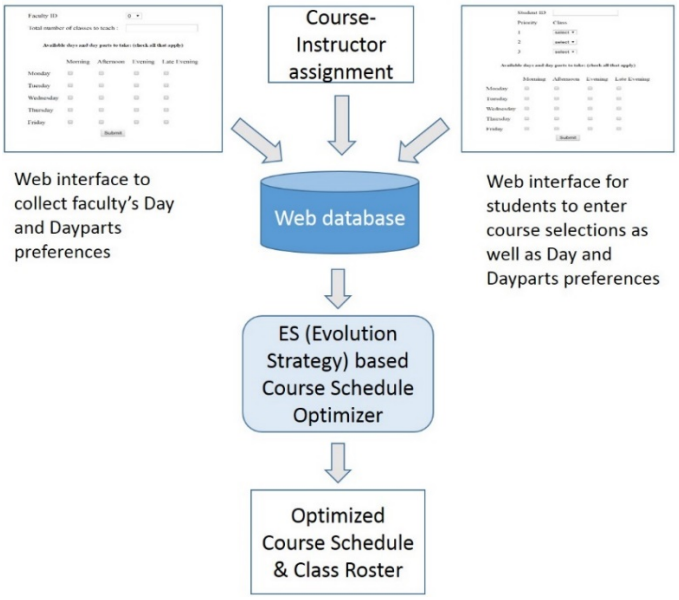
Lawrence Technological University
sjampani@ltu.edu

Introduction

Simplified course scheduling problem includes a finite set of courses C = {C1, C2, C3, ..., $C_{|C|}$}, time slots t = {t1, t2, ..., $t_{|t|}$}, students S = {S1, S2, ..., $S_{|S|}$}, professors P = {P1, P2, ..., $P_{|P|}$}, and a table which maps each course C to a professor P that teaches it. The optimization problem here is to find a best schedule that maximizes the number of course registrations while satisfying the following two constraints: (1) A professor cannot give more than one lecture at a time and similarly, (2) a student cannot attend more than one lecture at a time [1].

However, usually, course scheduling is done without applying optimization methods and not considering the faculty and students' preferences. This leads to the problem of less number of registrations to the university as well as the students are unable to enroll maximum number of courses in a given semester, thereby causing possible delays in the students' graduation. The main goal of this project is to generate the best schedule that maximizes the number of course registrations while satisfying faculty and students' preferences.

In this research, unlike previous system in [1], we propose a new scheduling framework that considers faculty member's preferred teaching day and day parts as well as students' preferred day and time to take selected courses. Since this is known to be a NP (Non-Polynomial) hard problem [2], instead of using traditional search algorithms, we decided to use evolutionary algorithms. The optimizer generates Optimized Course Schedule & Class Roster as shown in the following figure.

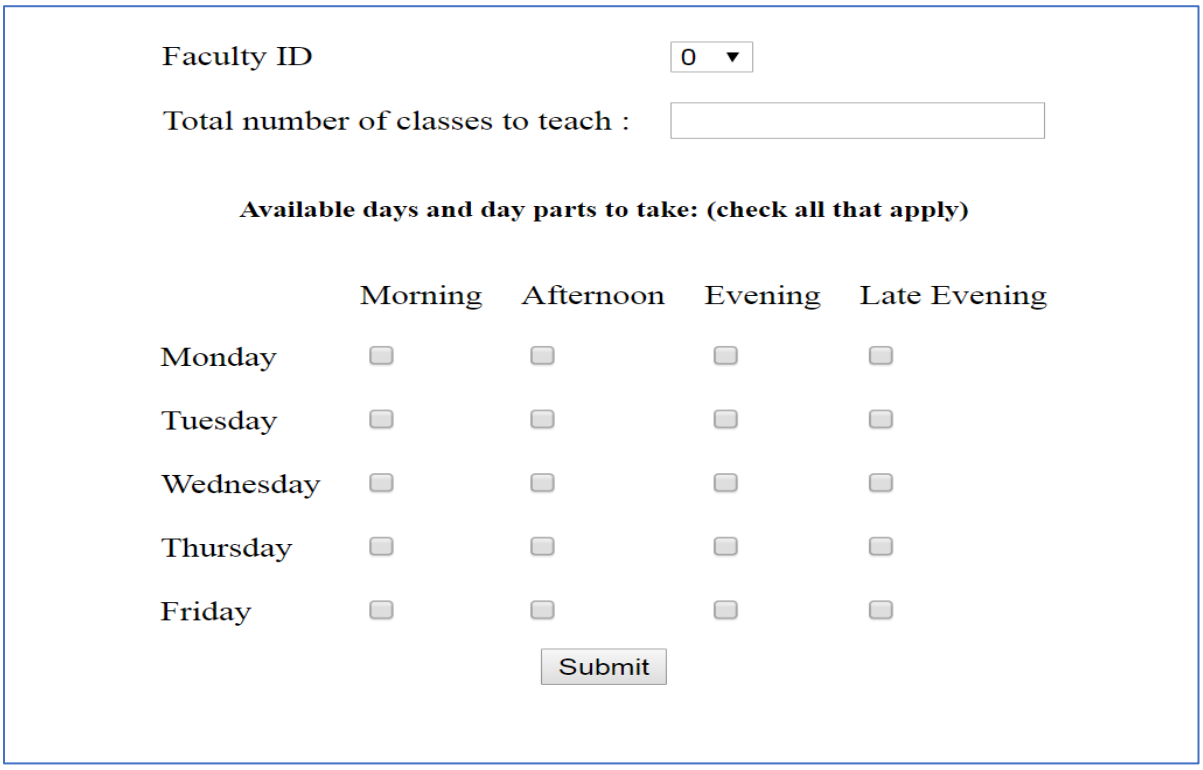


Methods

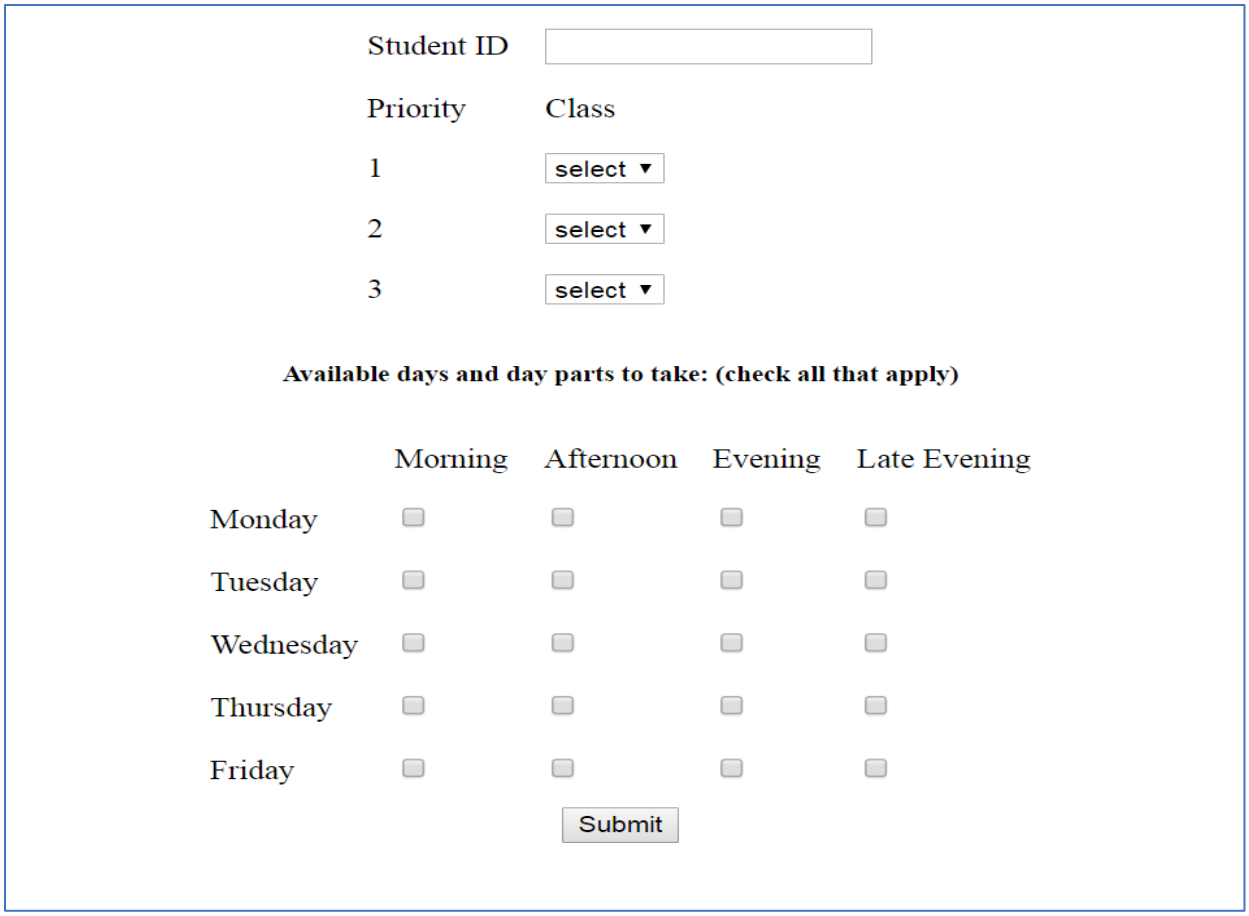
Web Interfaces to collect Faculty and Student Data

Two web interfaces are designed and implemented using JSP and Tomcat server to collect the preferences of the faculty as well as the students for their respective courses. Backend MySQL database was developed to store preferences as well as faculty-course data.

The Faculty Web Interface has the input field of “Faculty ID”, where faculty can select their ID from the dropdown menu and the field “Total number of classes to teach, C”, where the faculty can enter how many classes they are going to teach in the given semester. They can also select all the available dayparts in a week schedule. It is required for a faculty to check at least 3+C*2 check boxes. For example, if a professor teaches 3 classes, then at least 9 daypart check boxes must be selected.



In the student web interface, students enter their student id in the textbox of “Student ID” and select their top three priority classes from the dropdown menu. They can submit their preferences by selecting their available day and dayparts. Students may select only one check box or all the check boxes for the day and dayparts.



Optimization Algorithms Developed

Six algorithms using (N+N) Evolution Strategies [2] are designed and implemented to generate optimal schedules. The best schedule would be the one which can get the maximum number of enrollments. The following is the pseudo code for the simplest ES(N+N) algorithm with mutation operator only. Class-time for an offspring's course is selected using a uniform random number from the common dayparts between instructor's preferable dayparts and all students' preferable dayparts.

ES(N+N)

```
GET student and faculty preferences from database
Repeat T (Trial) times
   Generate & Evaluate N random parent schedules
   Repeat G (Generation) times
      For each parent schedules
         For each course in the parent schedule // mutation
            Randomly select a class-time considering both faculty and students' preference as well as scheduling constraints
         End-of-processing-all-the-courses
         Evaluate the offspring schedule to count the total number of registrations
      End-of-N-parent schedules
      Select N best schedules to be the next parent schedules from N (parents) + N (offspring) schedules
   End-of-G-number-of-Generations
   Find the best schedule from G generations
End-of-T-Trials
Find the best schedule from the T Trials
```

The above algorithm selects a time slot from all valid time slots using a uniform random number. As a simple change to the above algorithm, the following algorithm selects a time slot from the most popular dayparts from all students if the daypart is instructor's choice.

ES(N+N) with Highest Votes

```
GET student and faculty preferences from database
Repeat T (Trial) times
   Generate & Evaluate N random parent schedules
   Repeat G (Generation) times
      For each parent schedules
         For each course in the parent schedule
            Randomly select a class-time considering both faculty and students' preference with highest votes
            as well as scheduling constraints; // Mutation
         End-of-processing-all-the-courses
         Evaluate the offspring schedule to count the total number of registrations
      End-of-N-parent schedules
      Select N best schedules to be the next parent schedules from N (parents) + N (offspring) schedules
   End-of-G-number-of-Generations
   Find the best schedule from G generations
End-of-T-Trials
Find the best schedule from the T Trials
```

The above 2 algorithms employ only mutation operators. The following algorithm based on ES(N+N) introduces a simple crossover operator that exchanges a time slot information between parent and offspring schedules at a randomly chosen course.

ES(N+N) with Crossover

```
GET student and faculty preferences from database
Repeat T (Trial) times
   Generate & Evaluate N random parent schedules
   Repeat G (Generation) times
      For each parent schedules
         For each course in the parent schedule
            Randomly select a class-time considering both faculty and students' preference as well as scheduling constraints;
            Interchange a class-time between parent schedule and child schedule of a randomly chosen course
         End-of-processing-all-the-courses
         Re-Evaluate parent schedules to count the total number of registrations
         Evaluate the offspring schedule to count the total number of registrations
      End-of-N-parent schedules
      Select N best schedules to be the next parent schedules from N (parents) + N (offspring) schedules
   End-of-G-number-of-Generations
   Find the best schedule from G generations
End-of-T-Trials
Find the best schedule from the T Trials
```

The following algorithm introduces popular Dayparts with highest votes for mutation operation to the above ES(N+N) with Crossover algorithms.

ES(N+N) with Crossover & Highest Votes

```
GET student and faculty preferences from database
Repeat T (Trial) times
   Generate & Evaluate N random parent schedules
   Repeat G (Generation) times
      For each parent schedules
         For each course in the parent schedule
            Randomly select a class-time considering both faculty and students' preference with highest votes
            as well as scheduling constraints;
            Interchange a class-time between parent schedule and child schedule of a randomly chosen course
         End-of-processing-all-the-courses
         Re-Evaluate parent schedules to count the total number of registrations
         Evaluate the offspring schedule to count the total number of registrations
      End-of-N-parent schedules
      Select N best schedules to be the next parent schedules from N (parents) + N (offspring) schedules
   End-of-G-number-of-Generations
   Find the best schedule from G generations
End-of-T-Trials
Find the best schedule from the T Trials
```

The inventor of ES, Rechenberg suggested a simple rule called 1/5 success rule [3] in 1971 to improve the evolutionary search using the previous performance data in the past generations. The general idea of the 1/5 rule is to increase mutation rate if the mutation worked well (over 20% successes0 in the past *g* number of generations. If not, decrease the mutation rate. The following algorithm is an implementation of this idea to the ES(N+N) version.

ES(N+N) with 1/5 rule

```
GET student and faculty preferences from database
Repeat T (Trial) times
   Generate & Evaluate N random parent schedules
   Success_Rate = 0;
   Repeat G (Generation) times
      For each parent schedules
         For each course in the parent schedule // mutation
            If Success_Rate is better than 20%
               Randomly select a class-time considering both faculty and students' preference as well as
               scheduling constraints
            Else
               Randomly select a class-time considering both faculty and students' preference as well as
               scheduling constraints, only when tossed coin is head.
               (The chance of mutating the class-time is 50% -- decreasing mutation rate)
            End-If
         End-of-processing-all-the-courses
         Evaluate the offspring schedule to count the total number of registrations
      End-of-N-parent schedules
      Update Success_Rate;
      Select N best schedules to be the next parent schedules from N (parents) + N (offspring) schedules
   End-of-G-number-of-Generations
   Find the best schedule from G generations
End-of-T-Trials
Find the best schedule from the T Trials
```

The following algorithm introduces popular dayparts (with highest votes) for mutation operation to the above ES(N+N) with 1/5 rule algorithms.

ES(N+N) with 1/5 rule & Highest Votes

```
GET student and faculty preferences from database
Repeat T (Trial) times
   Generate & Evaluate N random parent schedules
   Success_Rate = 0;
   Repeat G (Generation) times
      For each parent schedules
         For each course in the parent schedule // mutation
            If Success_Rate is better than 20%
               Randomly select a class-time considering both faculty and students' preference with highest votes as well as
               scheduling constraints
            Else
               Randomly select a class-time considering both faculty and students' preference with highest votes as well as
               scheduling constraints, only when tossed coin is head.
               (The chance of mutating the class-time is 50% -- decreasing mutation rate)
            End-If
         End-of-processing-all-the-courses
         Evaluate the offspring schedule to count the total number of registrations
      End-of-N-parent schedules
      Update Success_Rate;
      Select N best schedules to be the next parent schedules from N (parents) + N (offspring) schedules
   End-of-G-number-of-Generations
   Find the best schedule from G generations
End-of-T-Trials
Find the best schedule from the T Trials
```

Before testing with large number of classes and students, we also developed an exhaustive search algorithm shown below to verify the correctness of the six ES algorithms with a small search space. In this algorithm, all the possible schedules are generated for given number of courses based on the faculty preferences.

Exhaustive Search

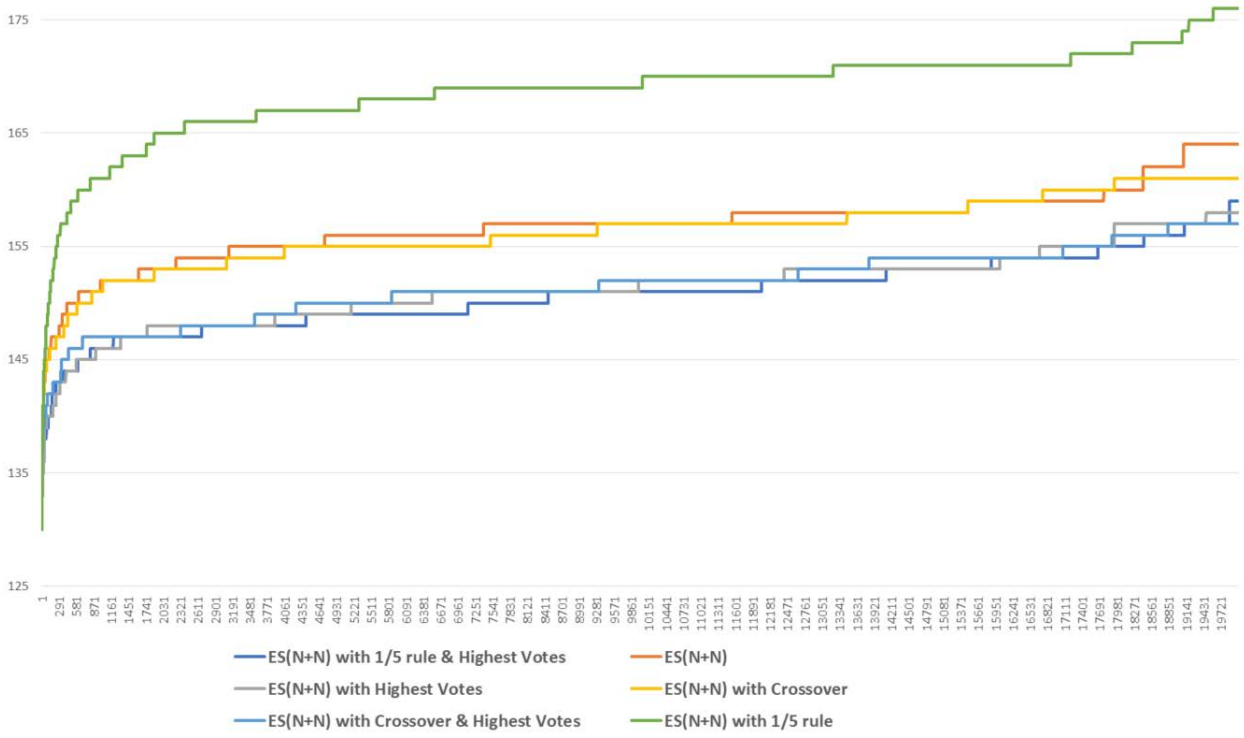
```
GET faculty preferences from the database
For all the available courses
   Permutate the faculty preferred timeslots with each course;
Generate all the possible schedules with the permutated lists using recursion
Evaluate all the generated schedules and find the first best schedule with the highest evaluation count
```

Test Methods and Results

In order to test our ES algorithms in a larger search space, the following parameter values were used. A program is written to randomly generate data for 400 students with three courses and three preferred daypart slots.

N (number of parents, number of offspring) = 10
T (number of Trials) = 20
G (number of generations in the evolution process) = 1000
C (number of courses) = 20
t (number of time slots) = 11

P (number of professors) = 11
S (number of students) = 400



The above graph summarizes the results of six Evolution Strategy algorithms. The X-axis of the graph represents the number of generations and the Y-axis represents the number of student enrollments. The best result was 176 enrollments from “ES (N+N) with 1/5 rule” algorithm as seen in green line in the graph.

We think best schedule of the generation one is equivalent to the manual approach that we are doing to prepare course schedules manually. The average value of the generation one from the six algorithms were 132.3. Since the best enrollment count from the six ES algorithms was 176, this research found an algorithm, ES (N+N) with 1/5 rule, that can achieve 33% improvement in course enrollment.

Summary & Conclusion

In this research, a course schedule optimization framework to include faculty and students’ preferred class time slots is introduced and six evolutionary algorithms are developed to find a valid schedule that satisfies time preferences of both instructor and students, eliminates time conflicts, and maximizes total number of enrollments for a University. Testing results clearly show that this system is applicable to find better schedules that increase the number of course registrations for a University. In addition, it is worthwhile to note that our system can simplify & revolutionize course registration processes, since it requires for students to specify only courses to take and his or her available day and dayparts. Our system will automatically assign courses to actual class time slot without time conflicts for each students.

Future work includes:

1. Considering multiple lectures per week. Current algorithm assumes only one lecture per week.
2. Handling courses with multiple instructors.
3. Considering online classes that does not have fixed meeting time.

4. Generation of a list of students who have classes that were not registered due to instructors' class-time preferences.
5. Considering class priorities set by students in generating course schedules.
6. Improving crossover operator since the current version is mating between a parent and its child. Traditional crossover operators mate between parent individuals.
7. Improving 1/5 rule algorithms since the current version is using only two cases of mutation rates when changing class time slot of a course, 100% or 50%.

Acknowledgement

Faculty Advisor: CJ ChanJin Chung
cchung@ltu.edu

References

[1] Tomas B. George, Vitaliy Opalikhin and ChanJin Chung, "Using Evolution Strategy for a University Timetabling System with a Web based interface to gather real student data", Proceedings of GECCO (Genetic and Evolutionary Computation Conference) 2003, Editor: Bart Rylander, Chicago, Illinois, July 12-16, pp. 107-113.
[2] Even, S., Itai, A., Shamir, A.: On the Complexity of Timetable and Multicommodity Flow Problems. SIAM Journal of Computing. Vol. 5. No. 4. (1976) 691-703
[3] Hans-Paul Schwefel: Evolution and Optimum Seeking: New York: Wiley & Sons 1995.