Evolution Strategies

Applying Evolution Strategies to a University Timetabling System

Thomas B. George  
GE Global Asset Protection Services  
25925 Telegraph Road, Suite 400  
Southfield, MI 48034  
thomas.george@gegapservices.com  
248-948-5351

ChanJin Chung  
Math & Computer Science Department  
Lawrence Technological University  
Southfield, MI 48075  
Chung@ltu.edu  
248-204-3504
Evolution Strategies

Applying Evolution Strategies to a University Timetabling System

Abstract

Determining the best, or near best timetable of lecture/courses for a university department, which optimizes enrollment, is a challenging problem. Described herein is a platform developed to accept student-generated data of course preferences, the university department curriculum, and resource constraints. The platform then generates the best or near best feasible schedule using an Evolutionary Computation Algorithm combining crossover and mutation for a fixed population. In addition, the platform permits manipulation of the population size, the initial mutation rate, the rule for varying the mutation rate, and the crossover point the platform order to examine the impact of these parameters and to determine the best values for this class of timetable problem. While this is an ongoing project, included is a sample run on an artificial data sample which shows the efficiency of the algorithm.

1 INTRODUCTION

Generating the best timetable consists of scheduling courses into appropriate timeslots and assigning resources such as instructors, classrooms, projection facilities, etc. In addition, consideration should be given to maximizing enrollment while ensuring students progress towards completion of their degree requirements. At LTU, this requires consideration of scheduling courses both during the day for full time students and also in the evening for working students. By optimizing within these constraints, the university can provide the most economical and efficient operation while satisfying student demands on the curriculum, a critical criterion in this competitive day.

Because this problem incorporates a search space that is too large to explore exhaustively, a evolutionary algorithm is the natural choice. In fact, the class of timetabling problems has been intensively studies and known to be NP-hard (Even, Itai and Shamir, 1976). Use of population reproduction with crossover techniques may locate the region in which the optimal solution resides, but fail to search it for the local maximum. Thus a combined algorithm using both crossover and mutation would seem to give us the best results.

Since selection of the best population size, mutation rate, mutation rate adaptive rule and crossover point remains more of an art than a science, we elected to develop a platform that will accept real course selection data from the student body, before a schedule is published, in order to generate the best schedule possible. Then, using both generated and real life data, the algorithm can be run repeatedly while varying different parameters to see which values give the best result most consistently. Because we expect the Math & Computer Science Department to run the final version of platform over the course of 1-2 weeks, generating several solutions, from which the best may be selected, we are searching for the parameters which give the best peak performance over a small sample as opposed to the average performance (Eiben and Jelasity, 2002).

Finally, to be sure that the evolutionary algorithm is not suffering from poor pseudo random number generation
effects, several different number generators, including a
genuine random number source have been incorporated
into the algorithm platform (Walker, RandomX library).
While the better number generators slow down the
program significantly in terms of processing time, the
algorithm’s efficiency measurement is based solely on the
number of evaluations required to obtain a solution.

The primary purpose of this project is to evaluate the
application of an evolutionary algorithm combining
crossover and mutation, using actual student survey input,
for solving university curriculum timetable problems. If
successful, it is our ultimate goal to establish a working
system that can be used at LTU and also be readily
adapted for use by other university departments.

The remainder of the paper provides a brief description of
the problem representation along with a description of the
algorithm. Some initial results of the test platform
operation on small simple timetable problems are shown,
however the majority of the analysis of test data remains
to be completed.

2 LTU MCS CURRICULUM

The LTU Math & Computer Science Department
Curriculum consists of over 90 courses, any combination
of which may be offered in a given a semester. Required
or core courses often must be offered both during the day
and during the evening to accommodate the student body
which consists of both full time students and working
professionals. However, it is not necessary to identify
core courses versus elective courses to the algorithm. The
requirements of the various degree programs force
students to select and prioritize courses that they will
enroll in.

While many courses require pre-requisites, the algorithm
makes no attempt to enforce these requirements. Rather
we rely on the mechanism for gathering data from the
students as governed by their advisor approvals.

2.1 HARD CONSTRAINTS

There are hard constraints that must be satisfied to
produce a valid schedule. The hard constraints for this
curriculum are:

- A course section may only be scheduled once (the
  schedule may consist of multiple periods on one or
  more days).
- A course section must be scheduled for the
  appropriate number of hours in a week, and must not
  exceed 4 consecutive periods.
- A student may not enroll in two classes that share any
  common periods (can't be in two places at once).
- A student may not enroll in more than x courses in a
  semester (6 has been used as a practical limit). The
  student, however, may elect to limit the personal
  maximum to a lower number.
- A limited number of available classrooms and/or
  professors exists.
- Students may indicate what periods they are
  available. If a course is scheduled in other periods,
  they will not enroll in the course.
- Some courses may be limited to certain available
  periods, for example: guest professor available only
  Tuesday & Thursday evenings.

In most cases, the algorithm can be implemented in a
manner so that invalid schedule permutations cannot be
selected. This is desirable in the case of hard constraints
but not for soft constraints.

2.2 SOFT CONSTRAINTS

In the case of soft constraints, violations affect the fitness
rating of the resulting schedule but do not invalidate the
schedule. In this example, we treat enrollment as an
overriding factor. However, given two or more schedules
with near equal enrollment, the schedule with the fewest
soft constrain violations is considered the most fit. The
soft constraints for this curriculum are:

- Maximize total enrollment for the semester.
- Maximize student enrollment in preferred courses.
- All other things being equal, schedules with the
  highest average enrollment are preferred. 5 classes
  with 20 students are preferable to 20 classes with 5
  students as the former maximizes revenue and
  minimizes operational expenses. Note that we have
  must maximize both enrollment and fulfillment of
  student preferences before we implement this
criterion.

2.3 FITNESS

The determination of a timetable's fitness rating is based
(in order of decreasing priority):

1. Maximizing the total enrollment.
2. Minimizing the number of preferred courses in which
   students are not able to enroll. (It is simpler to count
   violations of student preferences than to infer
   fulfillment).
3. Maximizing average enrollment per class.

Some fuzziness in evaluating criteria 1, 2 and 3 as a
composite value is desirable. Use of fuzzy logic here is a
natural step but in this initial implementation the
evaluation method only compares lower order criteria if
the previous criteria evaluated to equal.

2.4 REPRESENTING A SCHEDULE AS A
   CHROMOSOME

Care was taken to represent a schedule in a manner that is
easy to manipulate. Initially, implementing each schedule
as a list of available periods in which courses can be
inserted was considered. This implementation makes it
difficult to find course conflicts or reschedule courses. The actual Schedule implementation used is a list of courses, each of which contains its own schedule. For example, an extremely simple schedule might consist of MCS6513 and MCS 6123, both scheduled on Monday and Wednesday from 7pm to 9pm. This is represented in Figure 1.

<table>
<thead>
<tr>
<th>Schedule A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enrollment Total</td>
</tr>
<tr>
<td>Preference Violations</td>
</tr>
<tr>
<td>Average Course Enrollment</td>
</tr>
<tr>
<td>Boolean isCancelled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of Courses</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS6513</td>
</tr>
<tr>
<td>MCS6123</td>
</tr>
</tbody>
</table>

Figure 1. Sample Data Structure of Schedule

Note that the scheduled course periods are represented as strings consisting of pairs of characters. By convention, days are represented as M,T,W,R,F & S (no Sundays used). The second character indicates the hour scheduled starting at 8 AM and proceeding in alphabetical order. Hence, L is the 12th character in the alphabet and represents the 7PM to 8PM time-slot. The characters used is by convention only. The implementation does not rely on any ordered sequence of characters. This representation may not be the most efficient, but it does have the following advantages:

- The resulting schedule printout is human readable without translation/manipulation.
- Using only the ASCII character set, at least 52 potential periods per day can be represented using upper and lower case letters. This allows a 24 hour period to be readily broken down into 30 minute segments, which provides sufficient detail for scheduling courses in a timetable.
- Methods for manipulation and comparison of strings are built into many programming languages. In this case, Java was used which has the further advantage of immutability of Strings, making it much easier to ensure that operations on one schedule do not corrupt another schedule.

2.5 INITIALIZATION DATA

Initialization Data such as available periods, student period availability and course period restrictions are coded in a similar format which facilitate comparison operators and building of schedules.

The university curriculum data begins with the total number of classrooms/professors available and a simple list of all valid combinations of periods. [This method does limit the search space significantly and it is proposed to enhance the program to generate period combinations based on hard constraints.] Each course is then listed with an optional list of available periods.

Student data consists of listing each student ID and a list of classes selected by the student. An optional limit on the number of courses in which to enroll is planned as a future enhancement. Currently students are limited to a maximum of 6 courses for enrollment.

The order of the course listings, as submitted by each student, is considered significant. The first course is considered to have the highest priority, followed by the second course and so on. Validation of this data by the input mechanism is critical. In addition, the student may optionally list the periods on which he or she will be available to take the course. This data enables the algorithm to generate a true to life enrollment for each schedule which in turn is used to determine the fitness of the schedule.

Finally, initialization includes selection of the EC parameters. An example of the parameter selection and operating output GUI’s are shown in Figure 2.

Figure 2. Screen Shot of Scheduler
3 THE EVOLUTIONARY ALGORITHM

The algorithm uses a variation of ES(N+N) where the N is the selected population size and the mutation rate and adaptive mutation rate can be selected in advance by the user. The current implementation reads in the initialization data from the parameter selection on the main GUI and also from two text files. One file documents the university curriculum and constraints. The second file documents the student preferences and constraints for enrollment.

3.1 INITIALIZATION:
Generate N (10 is the default) Schedules to make up the population. Schedules contain every possible class assigned "legal" meeting times. Determine the fitness characteristics for each schedule based on total enrollment, student preferences fulfilled (students can only attend one class during a given period), and average enrollment per course. Note that while every schedule contains every course, courses with enrollment below a certain threshold (5 at LTU) are cancelled.

3.2 START EVOLUTIONARY LOOP

3.2.1 Crossover:
1. Allow each schedule to mate with another schedule. Mates are currently randomly selected with the following conditions:
   • There will be at least 1 child for each parent.
   • Each Schedule will be a parent.
   • Schedules cannot mate with themselves.
2. Generate offspring. For each mated pair, the parent with the best fitness rating contributes X% (50% default) of scheduled courses, selected randomly. The second parent contributes the remaining courses. Thus every child contains all possible courses.
   Note that a single crossover point is NOT selected. Genes are selected randomly from the first parent until the selected contribution percentage is fulfilled. This should result in more degrees of freedom in traveling the search space.

3.2.2 Mutation:
1. All of the children are subject to mutation (parents are not). Each course (gene) in a child has a (Mutation Rate) % chance of mutation. (Provision of a switch to allow mutation of parents may enhance the algorithm’s ability to locate local maxima.)
2. If mutated, a new legal set of class times is randomly selected from the pool and assigned.

3.2.3 Enrollment (for each child):
1. Enrollment for each course within a child schedule is determined based on the student survey information. Students are not allowed to enroll twice in the same course or to enroll in a course that does not fit their available time constraints. They schedule all courses in order of decreasing preference until either all selections have been evaluated or their maximum enrollment has been reached.
2. The courses are sorted in order of decreasing enrollment.
3. Courses are assigned rooms (professor limit is equally valid) for each period scheduled in order of decreasing enrollment. If no rooms are available, that course is marked cancelled.
4. Students re-enroll in the courses that are not cancelled to determine the final enrollment.
   Note: it is possible that enrollment would be better with another selection of course cancellations, but the student preferences would have been ignored in that case).
5. A count of preference violations and determination of average enrollment per class is now recorded in the schedule object for future comparison purposes.

3.2.4 Survival of the Fittest
1. Schedules are now sorted in order of decreasing enrollment.
2. The top N (10) schedules survive.
3. If the best schedule is better than those from previous generations, an "Improved" Counter is incremented and the best schedule is saved.
4. A generation counter is incremented.
5. If the generation is a multiple of the Window Size:
   • The mutation rate *= OneFifthRule (1.2 is the default) if more than 1 of 5 generations have shown improvement.
   • The mutation rate *= OneFifthRule (1.2 is the default) if fewer than 1 of 5 generations have shown improvement.
3.2.5 Loop Decision:

1. If the population has not improved for an arbitrary number of generations (10 * population size currently), then the loop exits. Otherwise it continues.

3.3 EXIT COMPUTATION:

At this point, the best schedule can be viewed on screen or saved as a text file for later analysis. Future changes and enhancements are currently under consideration for implementation during the summer of 2002.

4 INITIAL TEST DATA

The following information gives an example of the performance of the platform using a small data sample to validate the algorithm operation.

Other tests using very small data samples were previously run to ensure that the platform implements required hard and soft constraints properly.

4.1 INPUT DATA:

![Image of Selected Parameters]

The Curriculum Input Data consisted of 22 valid schedules, a 10 room limit, and 60 different classes (most without schedule restrictions).

The Student Preference Input Data was artificially generated and consisted of 1072 course selections by 122 different students. Because the input data was manually generated, there is a great deal of repetition in student preferences and course choices.

4.2 CONVERGENCE RATE:

Convergence is extremely fast for this size problem. This is probably due to the repetitive patterns within the student preference data. It is expected that the problem will be much more challenging when actual student data is evaluated. Note also that the algorithm must iterate a minimum of 10 * the population size without improved enrollment before it exits. To date this measure has appeared extremely conservative as seen in the following graph.

![Algorithm Performance Graph]

Figure 4. Algorithm Performance with Sample Data

4.3 GENERATED SCHEDULE:

Finally, here is a view of the generated schedule.

![Image of Generated Schedule]

Figure 5. Partial View of Generated Schedule
You will note that courses with enrollment below 6 are marked "cancelled" and do not contribute to the total enrollment figure. An additional sorting of the data would be helpful to put the course ID's in sequential order as opposed to order of decreasing enrollment value.

10 subsequent runs using the same parameters and input data yielded the following results:

<table>
<thead>
<tr>
<th>Maximum Enroll.</th>
<th>Total Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>166</td>
<td>250</td>
</tr>
<tr>
<td>167</td>
<td>250</td>
</tr>
<tr>
<td>164</td>
<td>100</td>
</tr>
<tr>
<td>165</td>
<td>150</td>
</tr>
<tr>
<td>165</td>
<td>200</td>
</tr>
<tr>
<td>164</td>
<td>100</td>
</tr>
<tr>
<td>164</td>
<td>150</td>
</tr>
<tr>
<td>165</td>
<td>200</td>
</tr>
<tr>
<td>143</td>
<td>250</td>
</tr>
<tr>
<td>142</td>
<td>150</td>
</tr>
</tbody>
</table>

Based on this we find a fairly good peak value of 167 but a large standard deviation 9.4 for this small sample, indicating multiple runs will be necessary to obtain the best results.

5 CONCLUSIONS

This paper presents an ES(N+N) algorithm approach to solving practical University Timetable Problems. The implementation permits significant experimentation involving changing population size, crossover point and adaptive mutation rate parameters as well as using different random and pseudo-random number sources. The system design accommodates use of a separate front end that will collect actual student data for future course enrollment preferences.

At this stage in the project, no conclusions have been reached regarding the efficacy of the algorithm or impact of different parameters and pseudo random number generators due to the limited data actually evaluated at this time. We have found that the algorithm converges rapidly to a near best schedule but does not always obtain the actual best schedule for small problems (soluble using an exhaustive search algorithm).

In the case of the presented sample data 22 valid schedules, a 10 room limit, 60 different classes (most without schedule restrictions) and 1072 course selections by 122 different students was used to generate a valid schedule. Standard parameters of 50% crossover point, 5% initial mutation rate and 1.2 adaptive rate were used with successful results as the algorithm converged quite rapidly to a solution with a maximum enrollment of 166.

Subsequent runs using the same data sample and parameters yielded encouraging results, as the convergence was quite rapid in every case. However the variation in solutions was quite large (standard deviation of 9.41 with a mean result of 158.8 for 10 runs. Thus, unless use of the other pseudo random number generators and selection of different parameters yields more consistent results, it will be necessary to run the program multiple times (more than 10) in order to be assured that a good solution has been found.

6 FUTURE WORK

Going forward, it is our intent to capture actual data for the next several semesters from the LTU student body for enrollment preferences in courses offered by the Math and Computer Science Department. This and generated data will be used to validate the algorithm and determine the best parameter values. Our evaluation criteria for algorithm performance will be based on both run time (measured in number of generations required to converge) and peak enrollment value for a small sample of runs (limited to what can be practically run within a two week period).

References