CS 461
Fall 2022
Program 2 – Genetic Algorithm

For this program, you will use a genetic algorithm to attempt to build a class schedule.

The University of Many Kinds of Computation (UMKC) has been attempting to develop a better method of scheduling their classes. There are many competing factors in scheduling classes, so a single ideal schedule may not be possible. (This is a simplified version of the real problem.)

- All of these courses are scheduled for MWF, for 50 minutes, so we need only determine the time slot, room, and instructor for each course.
- The School of Sophisticated Estimation (SSE) has several classrooms available and has full control of those rooms, and can put classes into any time slot they wish. (This is the single most fictitious part of the assignment.)
- Each faculty tends to teach from a small group of courses, but the same course isn't always taught by the same instructor, and the same instructor doesn't always teach the same courses. Thus, each course has a list of a few faculty who are preferred for teaching that course, and a few others who are able to teach the course. Any course can also be taught by any other faculty, but this is a stopgap if no other faculty are available.
    - You are given a list of all faculty. Use this to randomly assign course assignments and for random substitution (mutation). Use the preferred/other listing for scoring purposes, not for selecting faculty. (This keeps the generation/substitution process simple, and allows a broader exploration of the state space.)
- Each class has an expected enrollment. A class should be in a room big enough to hold the expected enrollment. Likewise, a room more than 3 times the expected enrollment can be used, but is too big, and there is a smaller penalty in that case.
- There are 2 sections (A and B) of some courses.
- Your program will assign, for each course:
    - Room
    - Time
    - Instructor
- Initially, this assignment will be random. You'll create a population of random possible schedules (N >= 500) and then apply a genetic algorithm to improve it.

Fitness function:

- For each class, fitness starts at 0.
- Class is scheduled at the same time in the same room as another class: -0.5
- Room size:
    - Class is in a room too small for its expected enrollment: -0.5
    - Class is in a room with capacity > 3 times expected enrollment: -0.2
    - Class is in a room with capacity > 6 times expected enrollment: -0.4
    - Otherwise + 0.3
- Class is taught by a preferred faculty member: + 0.5
- Class is taught by another faculty member listed for that course: +0.2
- Class is taught by some other faculty: -0.1
- Instructor load:
    - Course instructor is scheduled for only 1 class in this time slot: + 0.2
    - Course instructor is scheduled for more than one class at the same time: - 0.2
    - Instructor is scheduled to teach more than 4 classes total: -0.5

- Instructor is scheduled to teach 1 or 2 classes: -0.4
  - Exception: Dr. Xu is division chair and has other demands on his time. No penalty if he's only teaching 1 or 2 courses (or isn't on the schedule at all).
- If any instructor is teaching in consecutive time slots: Same rules as for CS 191 and CS 101 in consecutive time slots—see below.

Course-specific adjustments:

- The 2 sections of CS 101 are more than 4 hours apart: + 0.5
- Both sections of CS 101 are in the same time slot: -0.5
- The 2 sections of CS 191 are more than 4 hours apart: + 0.5
- Both sections of CS 191 are in the same time slot: -0.5
- A section of CS 191 and a section of CS 101 are taught in consecutive time slots (e.g., 10 AM & 11 AM): +0.5
  - In this case **only** (consecutive time slots), one of the classes is in Bloch or Katz, and the other isn't: -0.4
    - It's fine if neither is in one of those buildings, of course; we just want to avoid having consecutive classes being widely separated. Both on the quad (any combination of FH, MNLC, Haag, Royall) is fine, or one class in one of the 'distant' buildings and the other on the quad.
- A section of CS 191 and a section of CS 101 are taught separated by 1 hour (e.g., 10 AM & 12:00 Noon): + 0.25
- A section of CS 191 and a section of CS 101 are taught in the same time slot: -0.25

Computing the fitness function:

- The fitness function for a schedule is the sum of the fitnesses of the individual classes in it (i.e. add up the scores for each class using the above rubric, and sum the classes).
- Use **softmax** normalization to convert fitness scores into a probability distribution.[1]
- As you select pairs for reproduction, whether you produce 1 offspring per pairing or both possible offspring is an implementation decision.
  - Programming note: Access to the parent population is read-only; copy 2 parents, produce some offspring, add them to a next-generation pool. This can parallelize nicely.
- Use a mutation rate of 0.01; if an item is selected for mutation, use random selection.
  - Note: This is probably too high—see below.
- Run at least 100 generations. Continue until the improvement in average fitness for generation G is less than 1% improvement over generation G – 100. Report the best fitness from the final generation, and print the schedule to an output file. (Don't worry unduly about output formatting, but do try to make it legible.)
- Once your program is running, note the fitness you get, and then cut the mutation rate in half. As long as your results continue to improve, continue cutting the mutation rate in half until things appear to be stable.

---

1 Alternatively, rather than computing a probability distribution, you can keep your schedules in a minheap. The item at the top is the least-fit individual and is selected for removal. Select any 2 schedules at random from the heap as parents for a replacement. Find the fitness of the new offspring, put it at the top of the heap, and restore the heap property. This ensures we are always removing the least-fit individual. A more-fit individual will stay farther down in the heap longer, and thus have more opportunities to reproduce, than a lower-fitness individual which will find itself migrating to the top of the heap as breeding continues and the population gradually improves. In this case, a generation is N replacements, where N is the size of the population. (If you have a heap of size 500, then 500 remove/replace cycles constitute 1 generation.) This doesn't parallelize as nicely as the other method, but bypasses the need for normalization completely.

Other programming notes:

- The system-supplied random number generator will be fine for this project; there's no need to try to drop in an "improved" generator. (The generators in early compilers did have some definite weaknesses, but almost all modern languages now use the Mersenne Twister or a cryptographic or hash-based RNG, more than strong enough for our purposes.)

Turn in:

- Your source code: Python, C++, C#, Java, or Racket. (If your heart is set on using some other language, talk to me, but you're going to need a very good reason.)
- Sample output
- A short report (probably a page or so) in Word or LibreOffice format, discussing:
  - What were the challenges in writing the program? Or did it seem to go smoothly from the beginning?
  - What do you think of the schedule your program produced? Does it have anything that still looks odd or out of place?
  - How would you improve the program, or change the fitness function?
  - Anything else you feel like discussing, asking about, bragging about, etc.

Appendix: The Classes, The Rooms, and The Times
10 classes, 7 rooms, 6 times

Faculty: Gharibi, Gladbach, Hare, Nait-Abdesselam, Shah, Song, Uddin, Xu, Zaman, Zein el Din

CS101A, CS101B:
Expected enrollment: 50
Preferred instructors: Gladbach, Gharibi, Hare, Zein el Din
Other instructors: Zaman, Nait-Abdesselam

CS191A, CS191B:
Expected enrollment: 50
Preferred instructors: Gladbach, Gharibi, Hare, Zein el Din
Other instructors: Zaman, Nait-Abdesselam

CS201:
Expected enrollment: 50
Preferred instructors: Gladbach, Hare, Zein el Din, Shah
Other instructors: Zaman, Nait-Abdesselam, Song

CS291:
Expected enrollment: 50
Preferred instructors: Gharibi, Hare, Zein el Din, Song
Other instructors: Zaman, Nait-Abdesselam, Shah, Xu

CS303:
Expected enrollment: 60
Preferred instructors: Gladbach, Zein el Din, Hare
Other instructors: Zaman, Song, Shah

CS304:
Expected enrollment: 25
Preferred instructors: Gladbach, Hare, Xu
Other instructors: Zaman, Song, Shah, Nait-Abdesselam, Uddin, Zein el Din

CS394:
Expected enrollment: 20
Preferred instructors: Xu, Song
Other instructors: Nait-Abdesselam, Zein el Din,

CS449:
Expected enrollment: 60
Preferred instructors: Xu, Song, Shah
Other instructors: Zein el Din, Uddin

CS451:
Expected enrollment: 100
Preferred instructors: Xu, Song, Shah
Other instructors: Zein el Din, Uddin, Nait-Abdesselam, Hare

Room	Capacity
Katz 003	45
FH 216	30
Royall 206	75
Royall 201	50
FH 310	108
Haag 201	60
Haag 301	75
MNLC 325	450
Bloch 119	60

All rooms are available all time slots.

Class times:
10 AM
11 AM
12 Noon
1 PM
2 PM
3 PM