Introdunction to AI, Assignment 2

Sviatoslav Sviatkin, CS-05

December 2023

Hello, world!

I've been saving all my changes to a repository on github. If you want, you can write to me and I will give you access, because at the moment the repository is private.

Deep description of EA algorithm

Algorithm descripton

1. The Assignment class is used to setup input/output folders and read test case input files.
2. For each test case, an EvolutionaryAlgorithm instance is created with the list of words to place.
3. The run() method starts the evolutionary algorithm process:
    - An initial random population of Crossword solutions is generated
    - An initial random population of Crossword solutions is generated
    - Solutions are sorted by fitness
    - Selection, crossover, and mutation operators are applied to create new populations
    - This evolves the populations over generations to find better solutions
4. Once a good enough solution is found or max generations is reached, the best solution is saved and output is written.

Key aspects

- Crossword and Word classes represent the problem state
- An initial random population of Crossword solutions is generated
- Evolutionary operators like selection, crossover, mutation used to evolve populations

- Selection, crossover, and mutation operators are applied to create new populations
- This evolves the populations over generations to find better solutions

Selection

1. The population is separated into two groups:
   - best individuals (top 10% based on fitness)
   - rest of the individuals
2. The best individuals are directly copied over to the next generation. This elitism ensures the best solutions are preserved.
3. For the rest group, tournament selection is used to select parents for crossover.
   - Tournament selection randomly samples a few individuals and picks the best fitness.
   - This pressure biases selection towards fitter individuals.
4. Crossover between selected parents produces new child solutions
   - Uniform crossover is used - words swap positions randomly between parents.
5. The child solutions are mutated to introduce variation
   - Different mutation operators modify word placement with certain probabilities.
6. The best individuals and new mutated children form the population for next generation.

Crossover

1. Two parent solutions are selected using tournament selection.
2. A uniform crossover approach is used rather than single/multi-point.
3. A random crossover rate between 0 and 1 is chosen (default 0.5).
4. For each word/gene position in the parents' genomes:
   - A random number is generated between 0 and 1.
   - If less than the crossover rate:
     - The word coordinates/direction are swapped between parents.
   - Otherwise they are retained.

5. This results in two child solutions with words swapped at various random positions.+
6. The probabilities of applying these mutations is evenly weighted.

Mutation

1. After crossover, all the child solutions undergo mutation based on a mutation rate (default 0.5).
2. For each Word in a solution, a random number is checked against the mutation rate.
3. If under the mutation rate, one of several mutation operators is applied:
   (a) Change X coordinate
   (b) Change Y coordinate
   (c) Change both coordinates
   (d) Change direction
   (e) Change direction and X coordinate
   (f) Change direction and Y coordinate
   (g) Change direction and both coordinates
4. The probabilities of applying these mutations is evenly weighted.

The mutations introduce variability in the solutions. By randomly tweaking the placement of words, it allows the evolutionary algorithm to explore more of the search space and prevent pre-mature convergence on local optima.

The right level of mutation is crucial - too high can result in too much randomness, while too low reduces exploration. A balance is needed to ensure continued evolutionary improvements.

Fitness Function

The fitness function evaluates how good a candidate solution (crossword layout) is. Lower fitness is better. It works by penalizing violations of the crossword constraints:

1. Check if words go out of bounds:
   - Penalize by 100,000 per word if true
2. Check words for overlap:
   - Penalize by 100 per overlap
3. Check intersections:

  - Build a graph connecting intersecting words
  - Build a graph connecting intersecting words

4. Check collisions (words sharing edges):
    - Penalize by 20 per collision.
5. Check connectivity in intersection graph:
    - Penalize by 1,000 per disconnected subgraph (set of words not connected to others via intersections)

By combining various spatial constraints (bounds, overlaps, collisions) and factual constraints (letter matches), the fitness function encourages evolutions of boards with:

- Words fitting on board
- No collisions
- Intersecting words having valid connections
- All words connected in single graph

Solutions that satisfy these conditions will have fitness scores equal to 0. The goal is to minimize this value over generations by evolving populations via selection, crossover and mutation operators.

Specifics of variations

EA parameters

Population Size:

- _population_size - Size of the population in each generation (default 100)

Selection:

- _select_best - Takes top 10% fittest individuals
- _roulette_selection - Fitness proportional selection
- _tournament_selection - Tournament size 3

Crossover:

- _mutation_rate - Probability of mutation in child solutions (default 0.5)

General EA Control:

- max_generation - Maximum number of generations (stop criteria) (default 100000)

- max_tries - Maximum times to rerun EA from scratch (default 100)
- idle_generations - Generations without improvement before restart (length of words in crossword * 1000)

Fitness Weights:

- Out of bounds word - 100,000 penalty
- Overlapping words - 100 penalty
- Letter mismatch - 5 penalty
- Collision - 20 penalty
- Disconnected graph - 1,000 penalty

Statistics

Generations

Time