Universidad de La Sabana
Facultad de Ingenería

Felix Mohr & Sergio Amortegui
2026.1

Fundamentos de Inteligencia Artificial

- Sheet 3-

Exercise 1 (Sudoku with Arc Consistency and General Constructive Search - 10 Points)

1. Implement REVISE as by the given format.
2. Implement AC3 as by the given format.
3. Implement GET_BINARIZED_CONSTRAINTS_FOR_ALL_DIFF.
4. Implement GET_GENERAL_CONSTRUCTIVE_SEARCH_FOR_BCN. The returned search object must be a GENERALCONSTRUCTIVESEARCH initialized with the reduced root state w_0.
5. Implement GET_BCN_FOR_SUDOKU.
6. Upload your solution to Gradescope and make sure that you are able to solve the provided 9x9 and 16x16 Sudokus.

If you solve the given 16 x 16 Sudoku in less than 120 seconds, your code will be automatically enabled for competition on a more difficult board. Your time on that board will be shown in a public leaderboard, and if your code finds a solution in less than 10 minutes, you get a bonus of 0.2 to your exam grade.

Exercise 2 (Genetic Algorithms - Practice)

1. Implement a routine for a genetic algorithm that receives four parameters (for reproducibility make sure that the whole procedure is also seeded with an optional `RandomState` object):
   a) `init(n)`: creates a collection of `n` random individuals,
   b) `crossover(i1, i2)` that generates two children from the individuals `i1`, `i2`,
   c) `mutate(i)` that creates a (maybe) mutated copy of individual `i`,
   d) `population_size` that specifies how many individuals should be in the population at the beginning (and end) of each iteration.

   In each step, create a crossover for *every* pair of individuals in the current population, and apply mutation to every generated offspring (unless mutation is `None`).
2. Implement a non-trivial crossover (offsprings $\neq$ parents) for the route optimization setting that guarantees two valid offspring routes from two parent routes.
3. Implement a non-trivial mutation (after mutation $\neq$ before mutation) for route optimization.

4. Implement `run_genetic_search_for_gcs(gcs, timeout, random_state, population_size)`. This function receives a GCS-compatible problem instance (e.g., the provided TSP instance) and should return a high-quality route within `timeout` seconds. You must ensure that a solution is returned within this timeout (or earlier).

   Hint: Random restarts are recommended to reduce the risk of getting stuck in local optima.

   In Gradescope, you must find a solution with score below 2.45 for a 10-location instance within 60s. If you do so within 10s, your solution will be executed in a competition on a 100-location instance with timeout 5 minutes, and your name will appear on the leaderboard.

Exercise 3 (Optional: Why Some All-Diff BCNs Are Infeasible - 0 Points) **Optional challenge.** Consider three variables X_1, X_2, X_3 with binary domains

$$D(X_1) = D(X_2) = D(X_3) = \{0, 1\}$$

and an All-Diff constraint over all three variables.

1. Show formally why this BCN has no solution.
2. Briefly discuss what AC3 should conclude (feasible or infeasible) and why.