EvoLP.jl

A playground for evolutionary computation in Julia

Xavier F. C. Sánchez Díaz and Ole Jakob Mengshoel

Symposium of the Norwegian AI Society

Outline

- A brief overview of Evolutionary Computation
- The problem
- The framework
- An example
- Future work

Evolutionary Computation

It is a branch of Computational Intelligence (the older term for AI [the true AI, not statistical function approximators])[1] that solves optimisation problems using evolution-inspired algorithms.

[1]: browse AI Intro syllabus if you think otherwise

Evolutionary Computation

It is a branch of Computational Intelligence (the older term for AI [the true AI, not statistical function approximators])[1] that solves optimisation problems using evolution-inspired algorithms.

Shameless plug to my Oct. 2021 presentation: https://saxarona.github.io/project/evo-intro/

[1]: browse AI Intro syllabus if you think otherwise

Evolutionary Computation

It is a branch of Computational Intelligence (the older term for AI [the true AI, not statistical function approximators])[1] that solves optimisation problems using evolution-inspired algorithms.

Shameless plug to my Oct. 2021 presentation: https://saxarona.github.io/project/evo-intro/

"I want a tool for prototyping evolutionary solvers such that I can swap a thing or two and see how it affects the run"

[1]: browse AI Intro syllabus if you think otherwise

The problem

1. DEAP was old and very inefficient
   a. They sort the population to get top **k** when **k** << population
   b. It was for Python 2
2. Python is slow
   a. Vectorising (numpy/scipy) EC feels *unnatural*
   b. Conda is a mess
3. C++ is great but
   a. I did not want to deal with memory addresses (again)
   b. I hate objects. Functions are way cooler

Therefore, Julia.

There was *Evolutionary.jl*

- Inconsistent
- Very odd software patterns (a type for an iteration of an algorithm???)
- Documentation was lackluster
- Focused too much on solvers

EvoLP – The taxonomy

A computational unit for each *step* in the evolution process:

- Initialisation
- Selection
- Crossover
- Mutation
- Survival

EvoLP – The taxonomy

A computational unit for each *step* in the evolution process:

- Initialisation (generic)
- Selection (generic)
- Crossover (generic)
- Mutation (generic)
- Survival (algorithm dependent)

EvoLP – The taxonomy

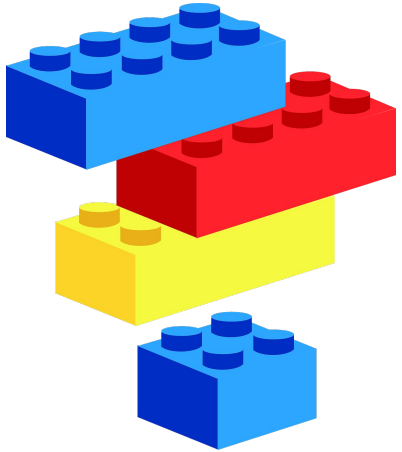
A computational unit for each *step* in the evolution process:

- Population Generators
- Selectors
- Recombinators
- Mutators

EvoLP – The taxonomy

A **block** for each *step* in the evolution process:

- Population Generators
- Selectors
- Recombinators
- Mutators



EvoLP – The taxonomy

A **block** for each *step* in the evolution process:

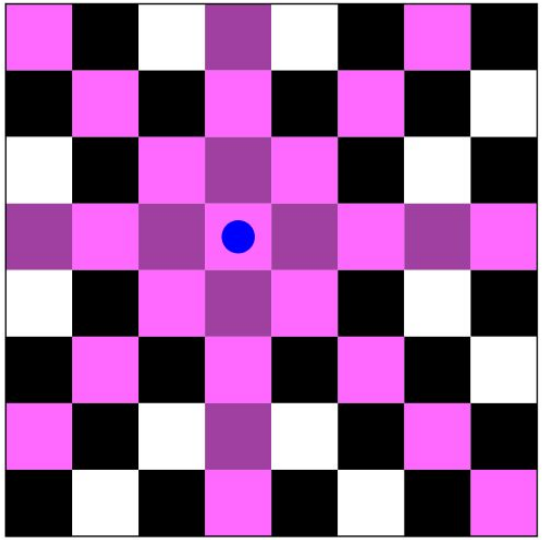
- Population Generators
- Selectors
- Recombinators
- Mutators
- Test functions
- Result reporting
- Statistics computing
- Built-in solvers

EvoLP – The taxonomy

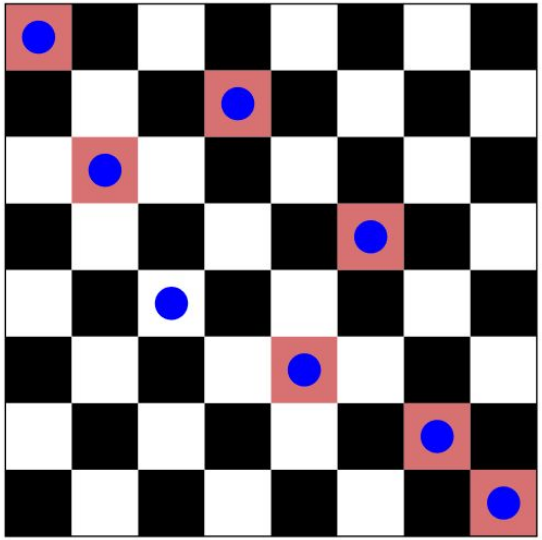
A **block** for each *step* in the evolution process:

- Population Generators
- Selectors
- Recombinators
- Mutators
- Test functions
- Result reporting
- Statistics computing
- Built-in solvers

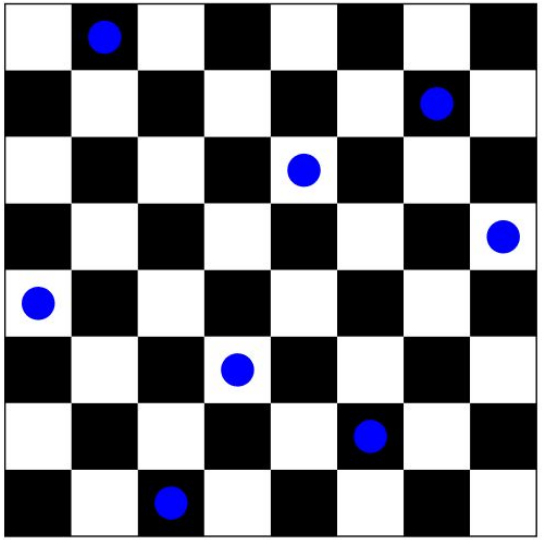
An example: 8-queens



Constraints of 1 queen



Conflicts of 8 queens

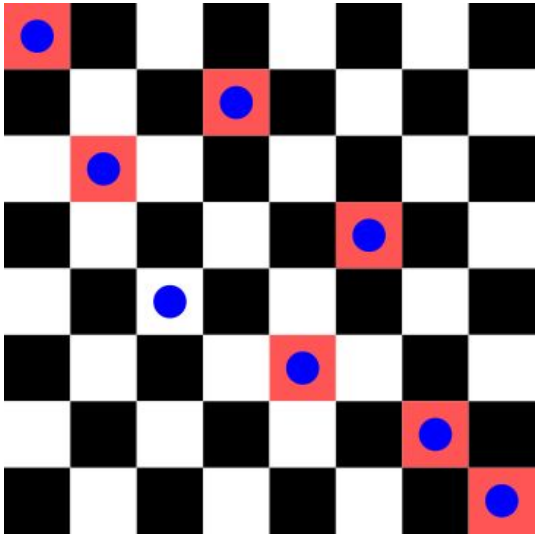


A solution

An example: 8-queens

A **block** for each *step* in the evolution process:

- Generator: random permutation
  - x = [1, 3, 5, 2, 6, 4, 7, 8]
- Selector: Random tournament
  - Tourney size of 5; choose 2
- Recombinator: OX1 crossover
- Mutator: Random swap

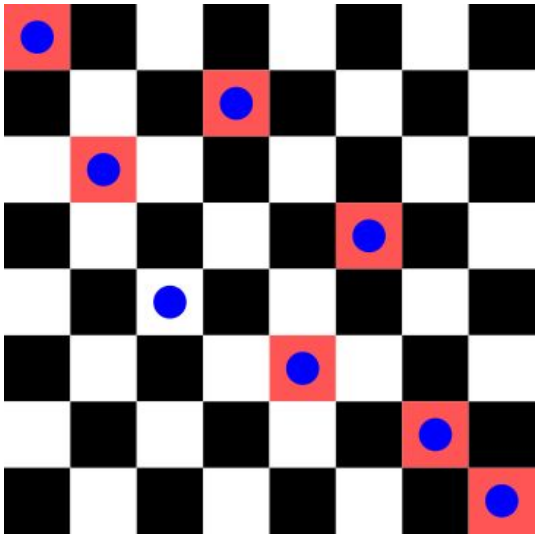


An example: 8-queens

A **block** for each *step* in the evolution process:

- Objective function: sum of conflicts
  - Σ_iDiagConstraints(q_i), i=[1..8]

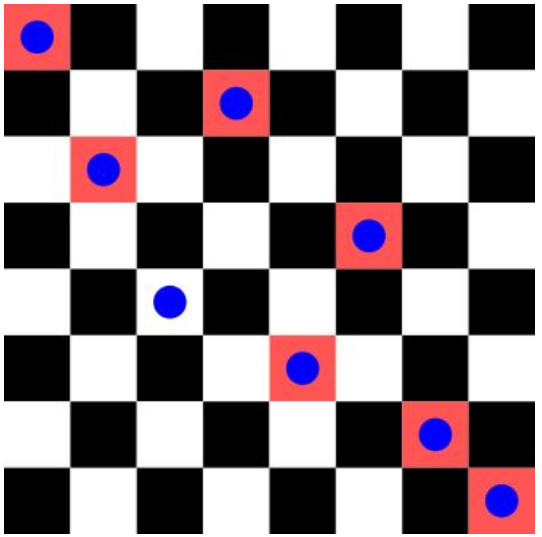
- Statistics computing
  - Max, min, mean and median of **f(x)**



An example: 8-queens

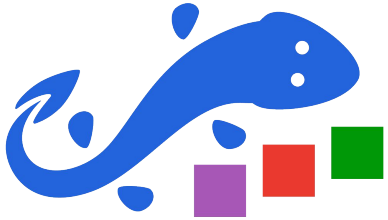
A **block** for each *step* in the evolution process:

- Custom solver: Steady GA
  - 2 offspring per generation
  - population size = 100
  - termination: 500 generations
  - Crossover probability: 1.0
  - Mutation probability: 0.8



EvoLP – Future work

- More examples
- Multi-objective support
- More test functions
- Parallelisation
  - Planning an MPI approach to simulate Islands on HPC clusters



EvoLP.jl

Thank you!

Project repo at https://github.com/ntnu-ai-lab/EvoLP.jl

Documentation at https://ntnu-ai-lab.github.io/EvoLP.jl/stable

```
julia> import Pkg
julia> Pkg.add("EvoLP")
```

Acknowledgements

- **Kristine Larssen** and **Benedikte Fiskergaard** for support and promotion
- **Kerstin Bach** for GitHub access and permissions
- **The IDUN HPC team** for access and computing resources

norwegian open ai lab