Genetic Algorithms

Description

This application uses a (somewhat) generic Genome class to solve different problems that can be solved by genetic algorithms. Different evolutionary algorithms are implemented. The applicatiion solves an equation and the queens problem with **n** queens.

Command line parameters are:

Parameter	Options
--type	**solver** - uses SolverGenome **queens** - uses QueensGenome
--strategy	**1+1** - uses (1+1) **u+l** - uses (μ+λ) **u,l** - uses (μ,λ) **rho** - uses (μ/ρ #λ)) **gen** - uses and abbrevaton of (μ/ρ #λ)
--qnum	(Optional) Takes positive int argument as number of queens used. (Default: 8)
--population	(Optional) Takes positive int argument as size for the starting population **μ**. (Default: 1000)
--children	(Optional) Takes positive int argument as number of children produced each generation **λ**. (Default: 2000)
--parents	(Optional) Takes positive int argument as number of parents used to produce one child. (Default: 2)
--generations	(Optional) Takes positive int argument as number of max Generations the algorithm runs. (Default: 10000)
--average	(Optional) Takes positive int argument as number of times the algorithm is run.
--performance	(Optional) Prints best fitness for each needed iteration.
--render	(Optional) If this parameter is set, the application renders the queens problem to an image.

Performance Measurements

Figure 1 plots the fitness over time for the different evolution strategies using the SolverGenome. Average of 100 runs each. As expected the (μ,λ) algorithm, which replaces all parents with children each generation, has a fluctuating fitness.

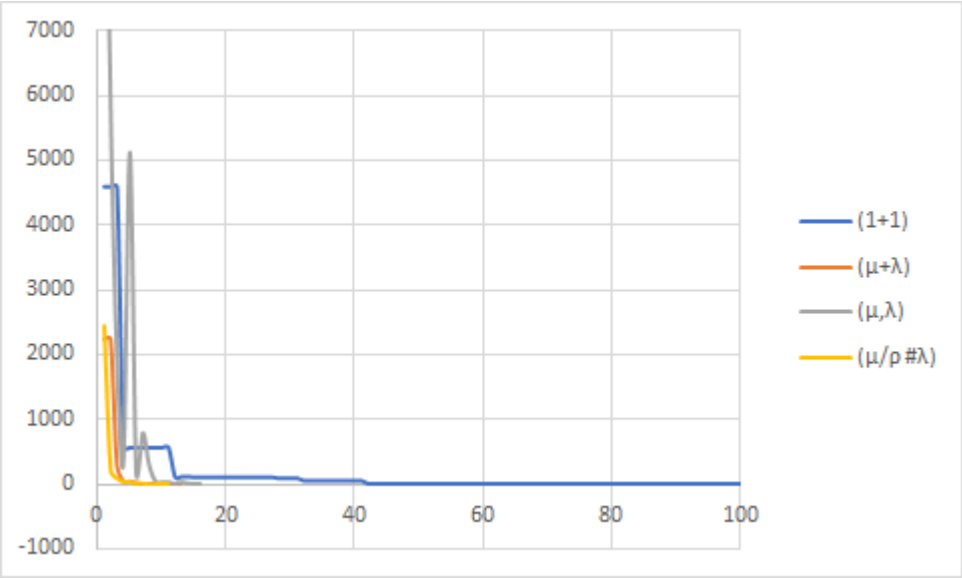

Figure 1

Figure 2 plots the fitness over time for different for the QueensGenome with different number of Queens. The Long periods of small values probably are local minima which don't have neighboring Genomes with lower fitness.

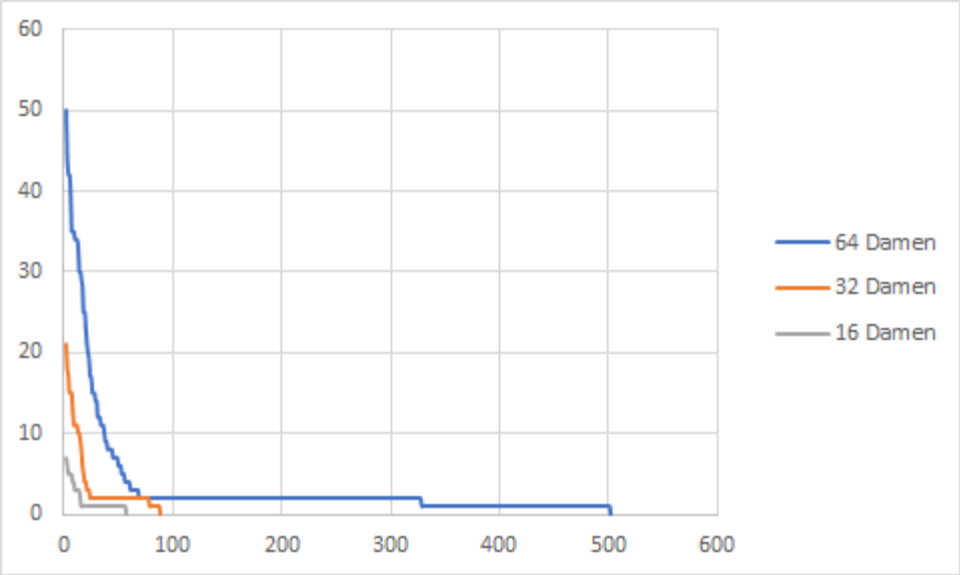

Figure 2