Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering
and Networks Laboratory

Bio-inspired Optimization and Design

Project 2: Knapsack Problem – Part II – Task 1

Discussion

Task 1 a)

a) Define a mutation operator for the knapsack problem. Note: you can make use of the neighborhood.

- Properties
  1. Every solution can be generated from every other solutions by means of mutation with a probability greater than 0.
  2. *d(x, x’) < d(x, x’’) => Prob[mut(x) = x’] > Prob[mut(x) = x’’]*
- Methods:
  - Random bitflips: $x_{i+1} = 1-x_i$ with p, $x_{i+1} = x_i$ with (1-p)

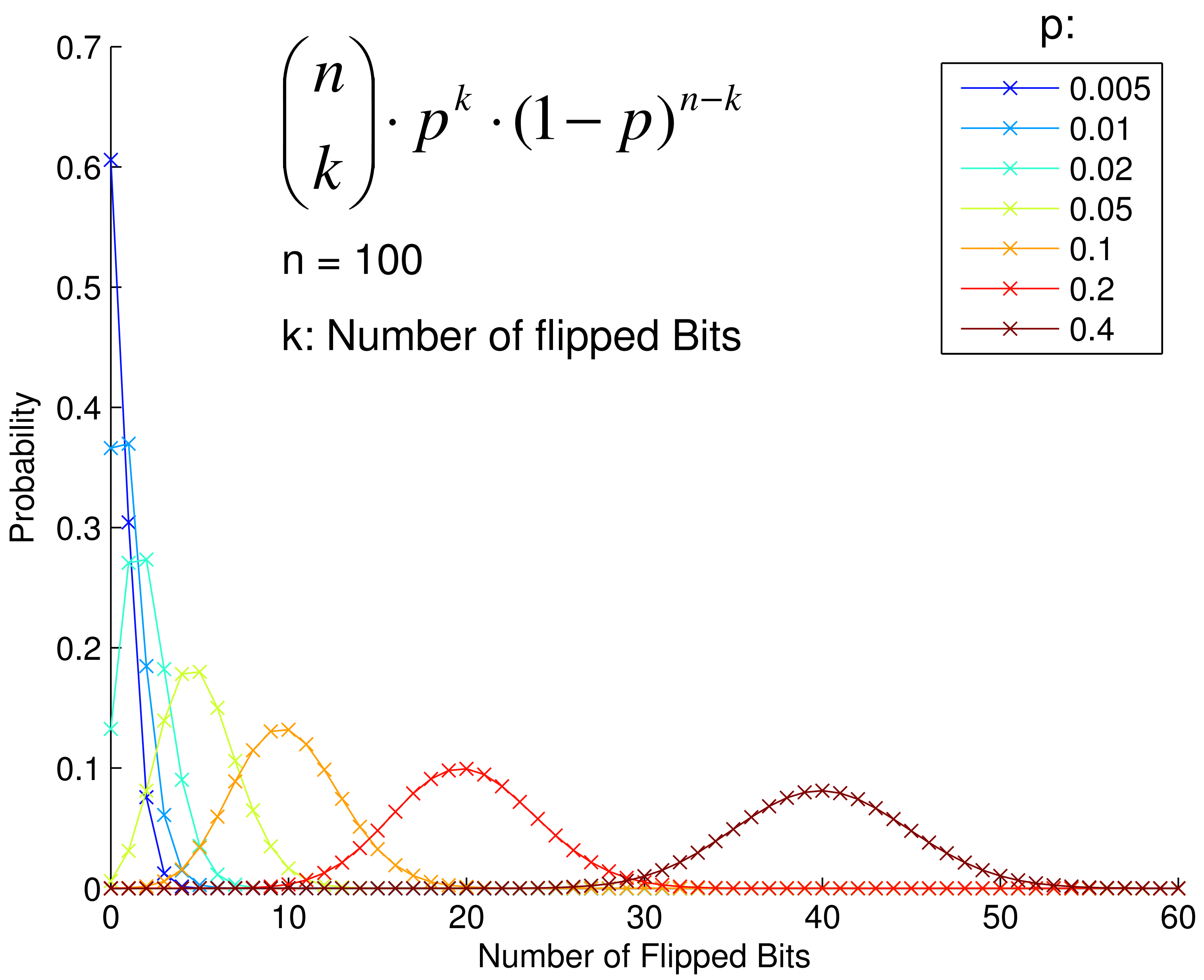
    -> P[k mutated bits] = $\binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$

Binomial Distribution

$$\binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$$

n = 100

k: Number of flipped Bits



Task 1 a)

a) Define a mutation operator for the knapsack problem. Note: you can make use of the neighborhood.

- Properties
  1. Every solution can be generated from every other solutions by means of mutation with a probability greater than 0.
  2. *$d(x, x') < d(x, x'') \Rightarrow Prob[mut(x) = x'] > Prob[mut(x) = x'']$*
- Methods:
  - Random bitflips: $x_{i+1} = 1-x_i$ with p, $x_{i+1} = x_i$ with (1-p)

    -> P[k mutated bits] = $\binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$

    E[mutated bits] = n*p

  - Neighborhood function: Choose d randomly
    - uniform sampling: 2nd property violated
    - better: exponentially decreasing sampling

Task 1 b)

b) Define a recombination operator, which produces two offspring solutions from a given pair of parent solutions.

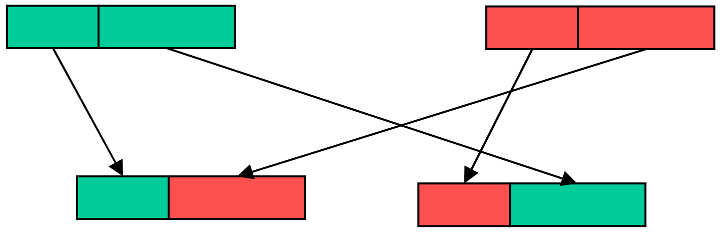
- Property:

  $x'' = recomb(x, x') \Rightarrow d(x, x'') \leq d(x, x') \, □ \, d(x', x'') \leq d(x, x')$:

- Methods:
  - k-point crossover

    -> Not a priory middle of bitstring

  - uniform crossover



1-Point-Crossover

b) The adopted recombination operator is the one point crossover recombination operator. It satisfies the property:
x'' = recomb(x, x') => d(x, x'') ≤ d(x, x') ∧ d(x', x'') ≤ d(x, x') as if we partition the bit encoding vector is two parts (part1 and part2) one taken from the father (part1(x)) solution and the other from the mother solution (part2(x')) then:
x'' = part1(x) U part2(x'))

d(x,x'') = d(part1(x),part1(x'')) + d(part2(x),part2(x''))
= d(part1(x),part1(x)) + d(part2(x),part2(x'))
= 0 + d(part2(x),part2(x')) <= d(x,x')

and

d(x',x'') = d(part1(x'),part1(x'')) + d(part2(x'),part2(x''))

= d(part1(x'),part1(x)) + d(part2(x'),part2(x'))
= d(part2(x'),part2(x)) + 0 <= d(x,x')

Task 1 c), d)

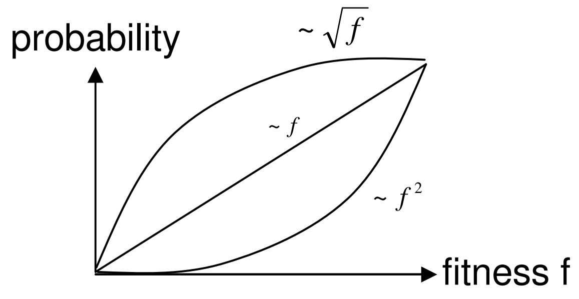
c) Define a mating selection operator.

d) Define an environmental selection operator.

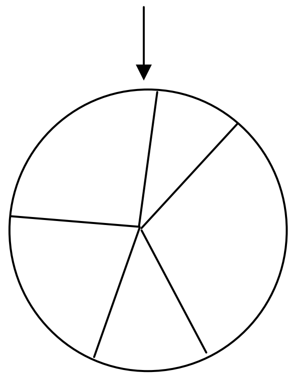
- Two types of selection schemes can be distinguished:
  - ❶ stochastic selection consisting of
    - sampling rate assignment
      Q_i = probability that individual i is chosen
    - sampling
      choose N individuals according to their sampling rates
  - ❷ deterministic selection

Mating Selection: Stochastic Sampling

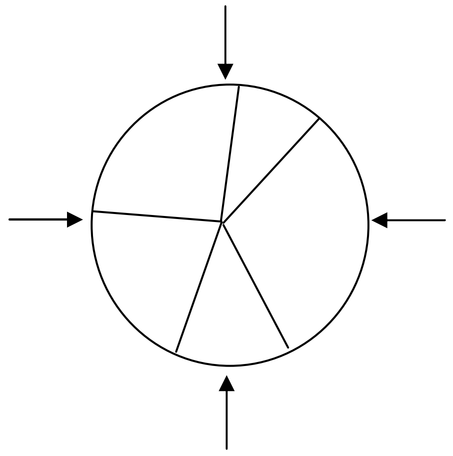
- Tournament Selection
- Fitness Proportional Selection
  - Assign each solution a probability



Roulette:



SUS:



  - Choose random element, select it with given probability

Environmental Selection: Deterministic Sampling

- Problem:

  Given: μ parents, λ offspring -> select μ new parents
- Methods:
  - Comma selection: {μ, λ} -> replace parents with μ best offspring

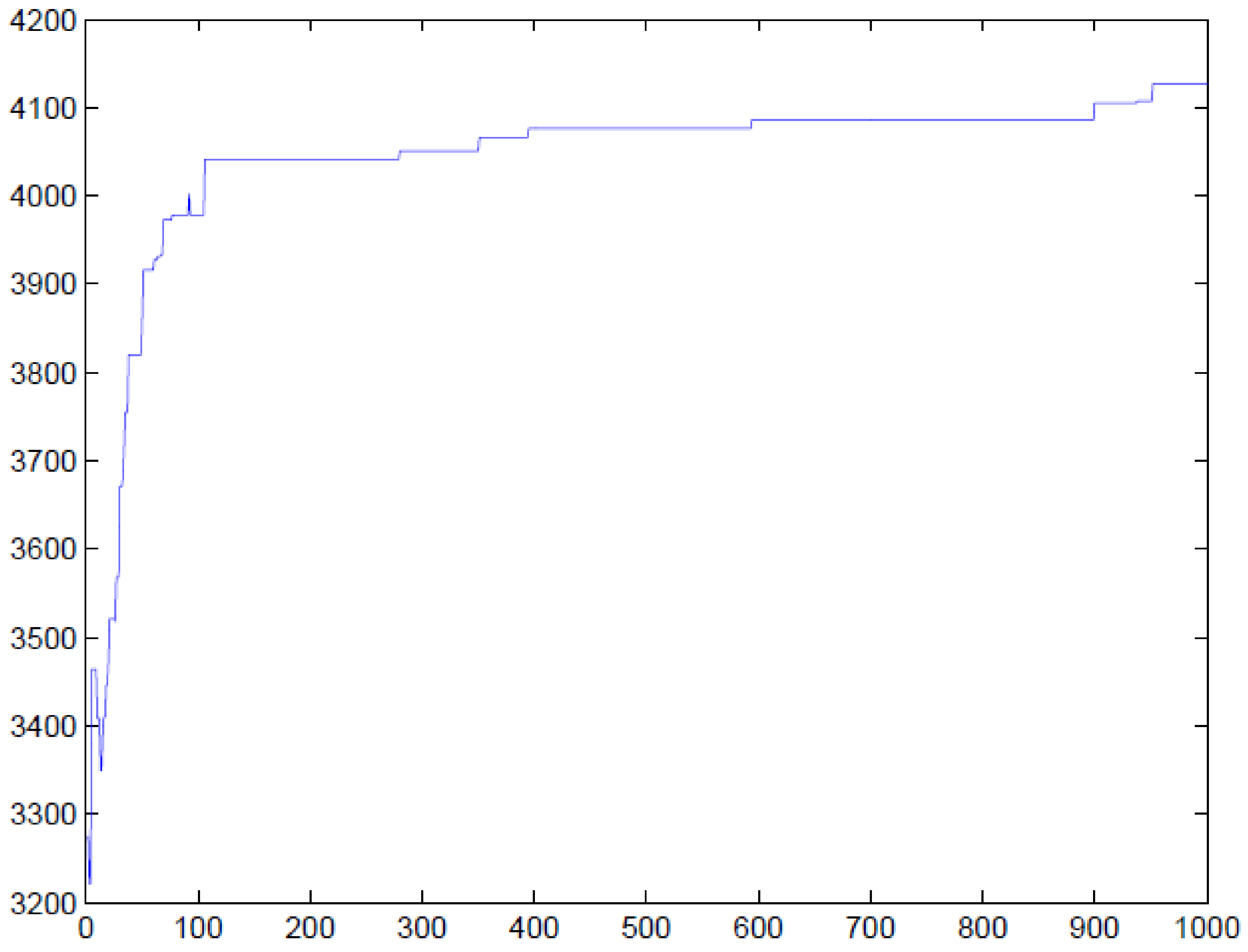
    -> select μ out of λ offspring
  - Plus selection: {μ + λ} -> select the μ best individuals from the μ old parents and λ individuals

- Comments:
  - Implicit comma selection (if μ = λ)
  - 50% best parents + 50% best offspring

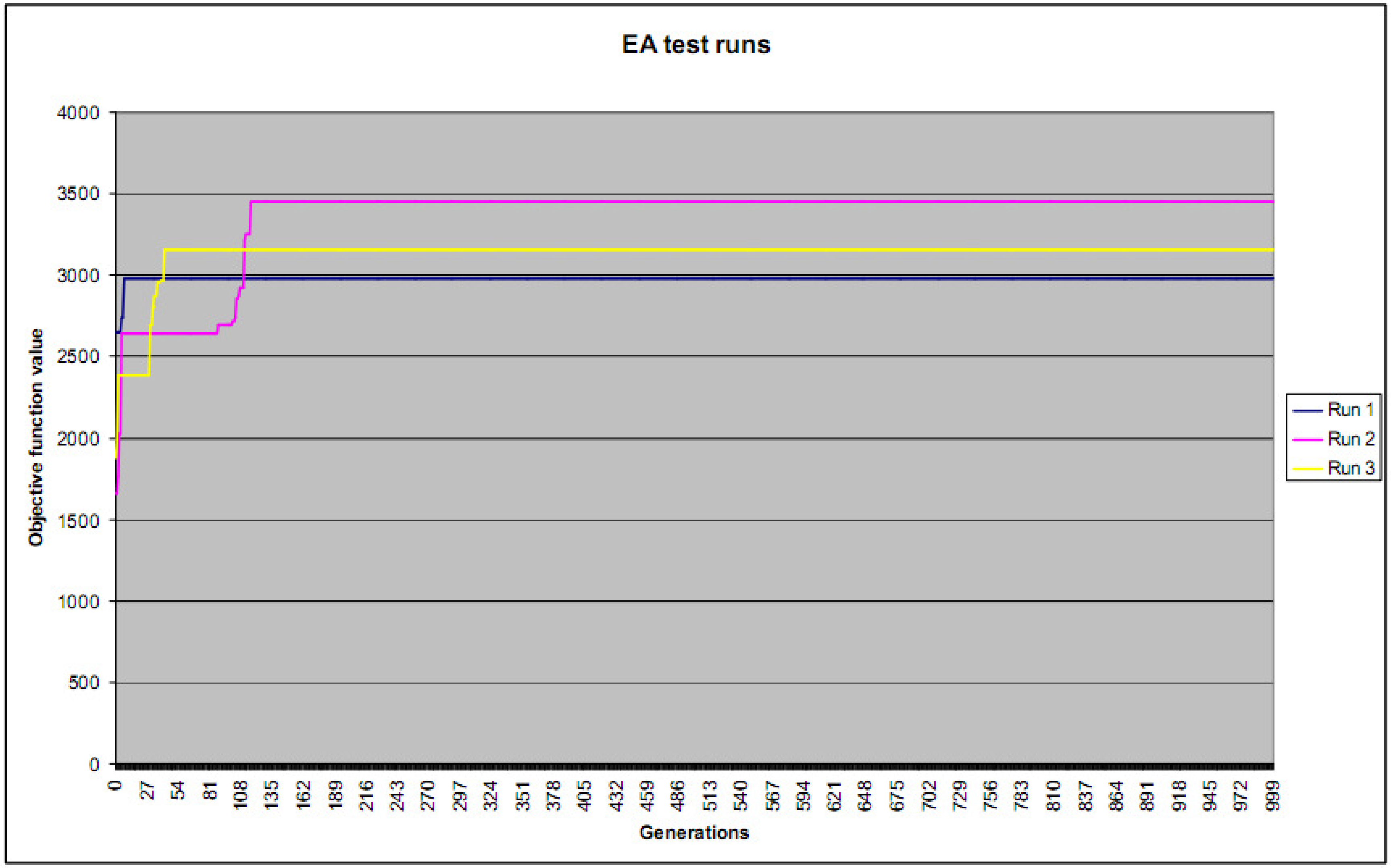
Task 1 e)

e) Implement an evolutionary algorithm with population size 100 and the operators you defined in a)– d). Use the objective function from project 1 as the fitness function. Run the algorithm on the problem instance given on the lecture website for 1000 generations. Create a plot to show how the best objective function value of the population develops over time. To that end, plot the number of generations versus the best objective value found so far.

Axis Labeling

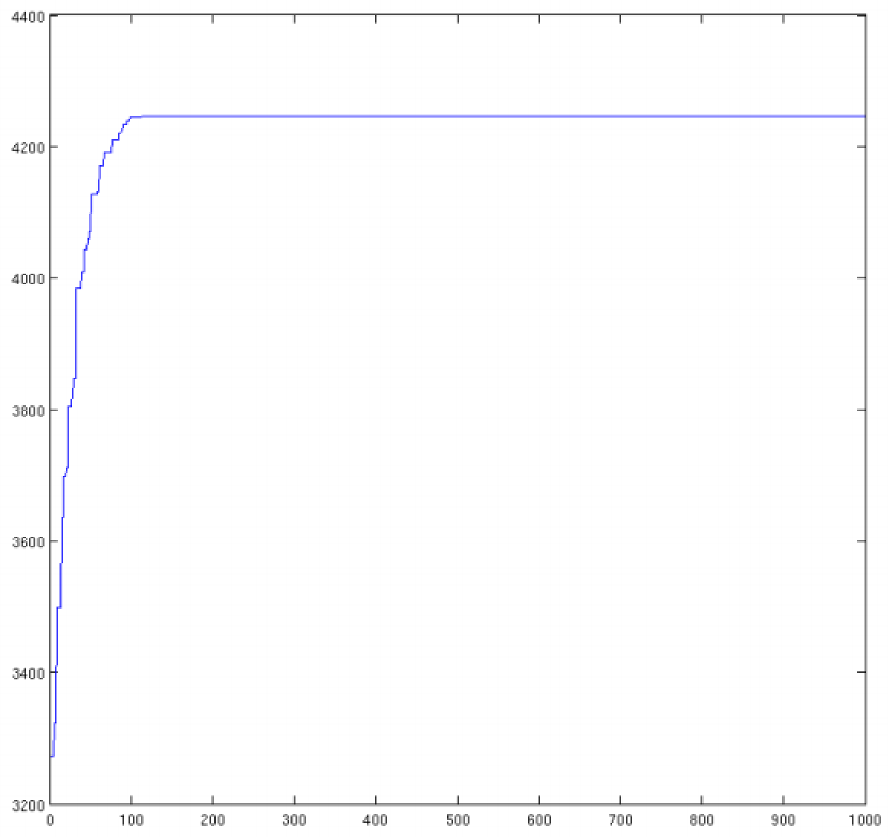


Local Optima

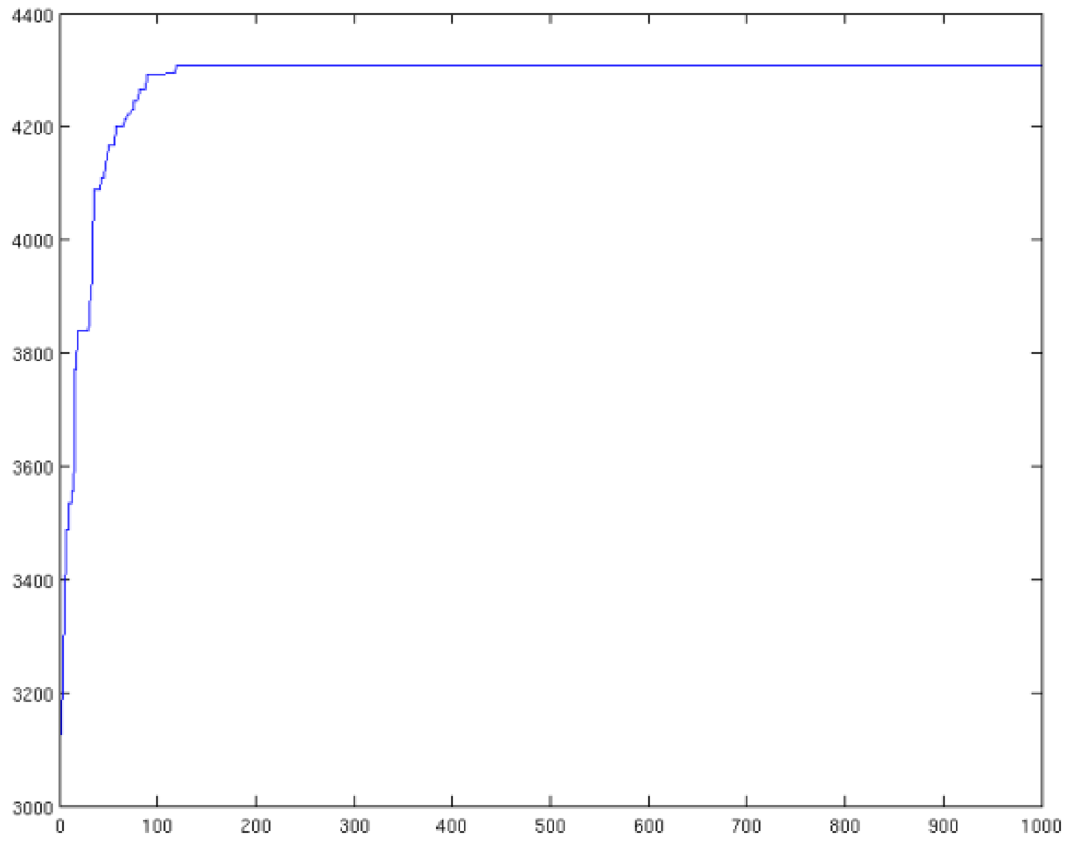


Is there a difference?

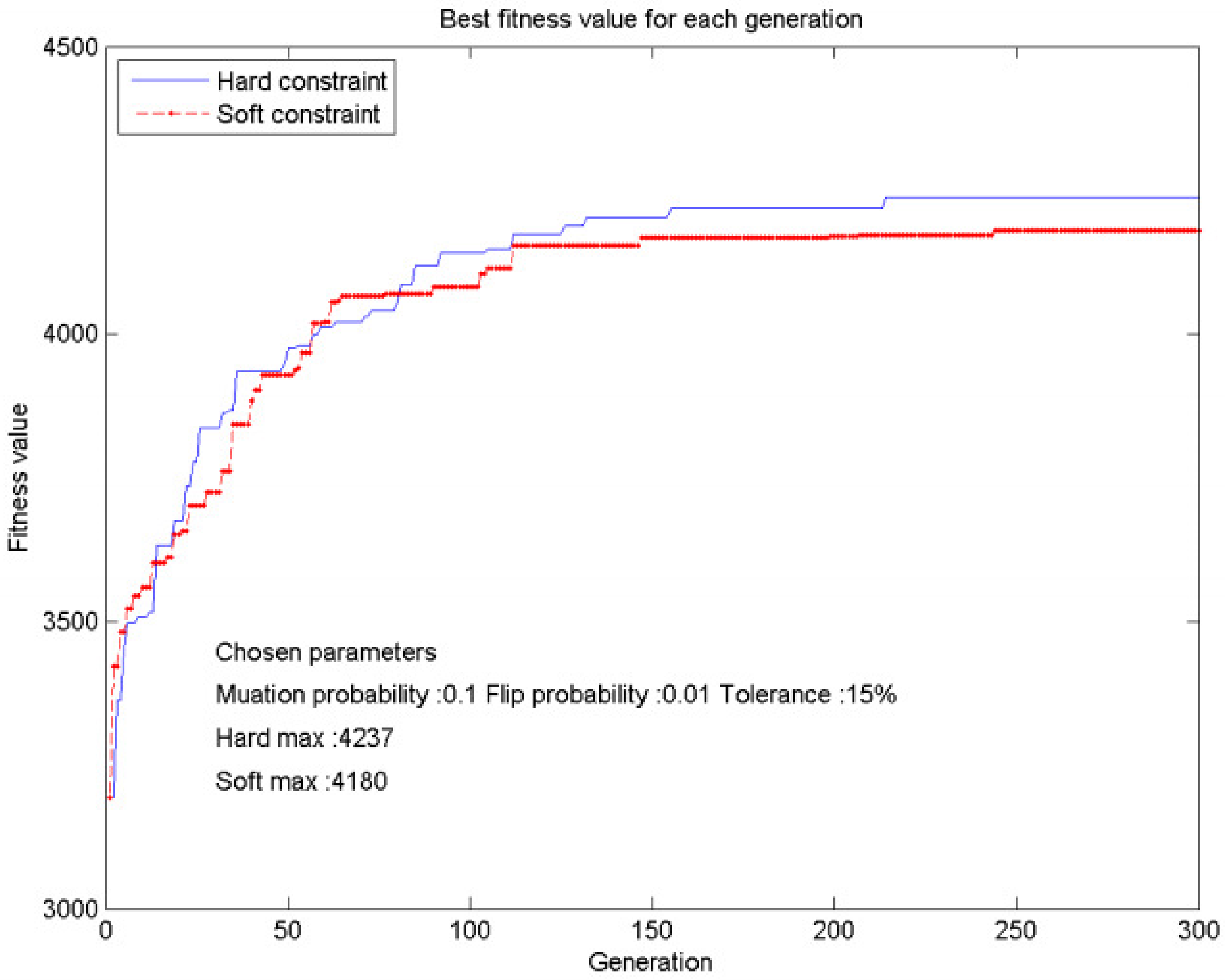
e)



g)



Better…



Chosen parameters

Muation probability :0.1 Flip probability :0.01 Tolerance :15%

Hard max :4237

Soft max :4180

Task 1 f)

f) How would you classify the constraint handling technique used so far? Suggest an alternative constraint handling technique and describe how you would have to change the algorithm accordingly.

- So far you set the objective function to 0 if the items exceed the maximum weight.
- Possible constraint handling techniques:
  - Penalize infeasible solutions, e.g., by substracting the (scaled) overweight
  - Ensure that only feasible solutions enter the population (all operators have to be adjusted!)
  - Repair / Healing: Throw away items with worst profit/weight ratio ⇨ diversity is lost
  - Use "multi-objectivization" by handling the constraint as additional objective (⇨ next week)

Task 1 g)

g) Run the modified algorithm on the same instance of e) and produce a similar plot. What differences do you see?.

- Often, the improved algorithm performed better
- What should you do if not?
  - Experience very useful, when using EAs
  - Change operators (mutation, recombination)
  - Change selection schemes (mating, environmental)
  - Many parameters are sensitive: do parameter optimization
  - Idea: adaptive parameters!