Evolutionary Algorithms: Dynamic Optimization Problems

Jiří Kubalík
Department of Cybernetics, CTU Prague

http://cw.felk.cvut.cz/doku.php/courses/a4m33bia/start

Dynamic Optimization Problems: Motivation

Dynamic optimizations constitute an important research area as a significant part of real world problems are problems subjected to dynamic and uncertain environments (scheduling, manufacturing, trading, etc.).

Dynamic Optimization Problems (DOPs) – a class of problems whose specifications commonly termed as *environment*

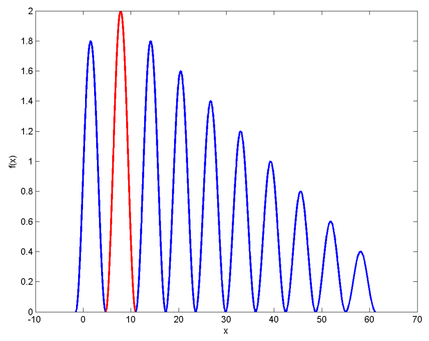
- optimization objectives, and/or
- problem-specific constraints, and/or
- environmental parameters and problem settings

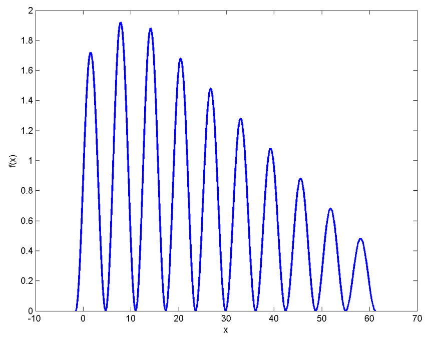
change over time, resulting in continuously moving optima.

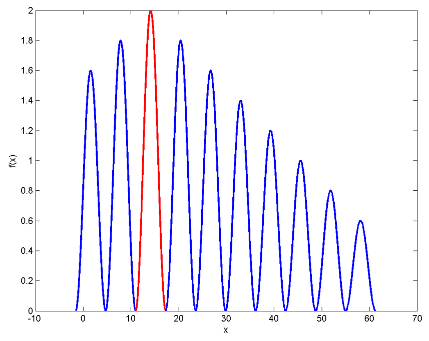
Ex.: Stochastic arrival of new tasks, machine faults, climatic change, market fluctuation, economic and financial factors.

The goal of optimization in dynamic environment is

- to continuously track the optimum, or
- to find robust solutions that perform satisfactorily even when the environmental parameters change.







Dynamic Optimization Problems: Characteristics

DOPs characteristics

- **Speed of change**
  - slowly changing environment
  - rapidly changing environment
- **Severity of change**
  - the environment changes only slightly
  - severe environment changes are observed
- **Periodicity of change**
  - cyclic dynamic environment – the environment returns to some states already observed in the past
  - random dynamic environment – changes at random.

Evolutionary Algorithms

Evolutionary Algorithm – due to its population-based search seems to be a natural candidate to solve dynamic optimization problems.

Problem with standard EAs is that the population eventually converges to an optimum and loses the diversity, consequently lose their ability to

- efficiently explore the search space, and
- adapt to a change in the environment.

Several ways to deal with the convergence problem

- **React on changes** – As soon as a change in the environment has been detected, explicit actions are taken to increase diversity of the population to allow the shift to the new optimum.
- **Maintaining diversity throughout the run** – Convergence is avoided all the time; it is hoped that a spread-out population can adapt to changes more easily.
- **Memory-based approaches** – EAs supplied with a memory to be able to recall useful information from the past generations.

React on Changes

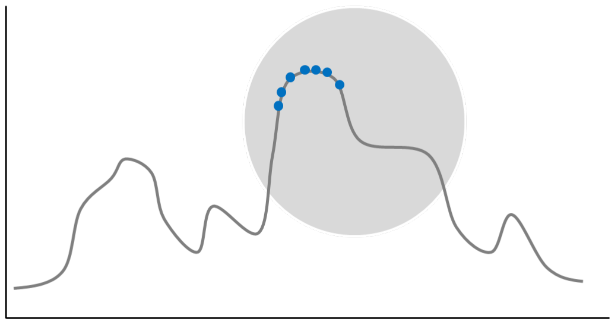
Change detection

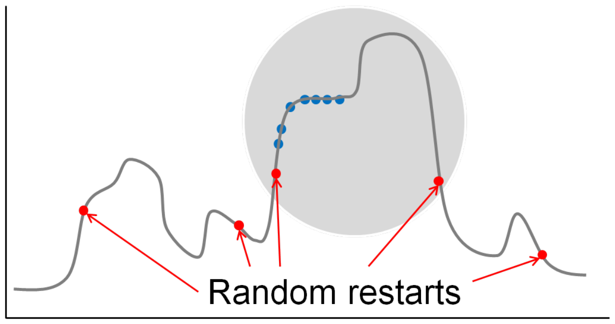
- The change points are known – the environment changes according to known scheme.
- Population-based detection – all members of the external memory are re-evaluated at the end of every iteration and the change of the environment is detected if the fitness of at least one memory member has been changed.

  Alternatively, just re-evaluate the best best-of-previous-generation point.
- Sensor-based detection – uses measurements of the landscape's fitness on prescribed points (sensors).

Restarted search

- The simplest way to deal with changes.
- Inefficient – does not utilize any knowledge acquired during the previous populations.





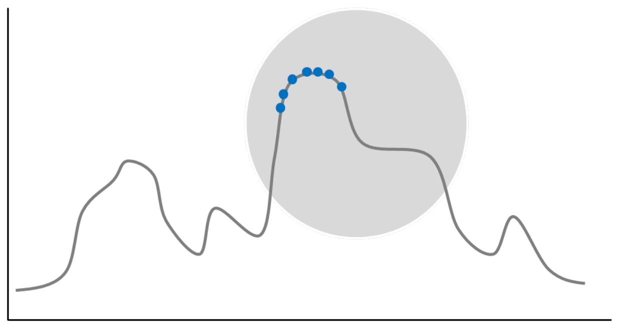
React on Changes

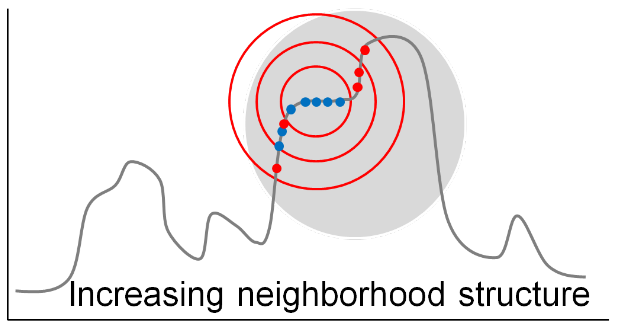
Hypermutations – increases population diversity by drastically increasing the mutation rate for some number of generations.

- All solutions in the population are affected.

Variable Neighborhood Search – uses a set of different neighborhood structures that are ordered in an ascending manner according to increasing neighborhood size. Whenever the local search fails to find better solution in the neighborhood of the current solution, a bigger neighborhood structure is tried.

- When used for DOPs, VNS swaps to larger neighborhood structure gradually after a change in the environment has been detected.



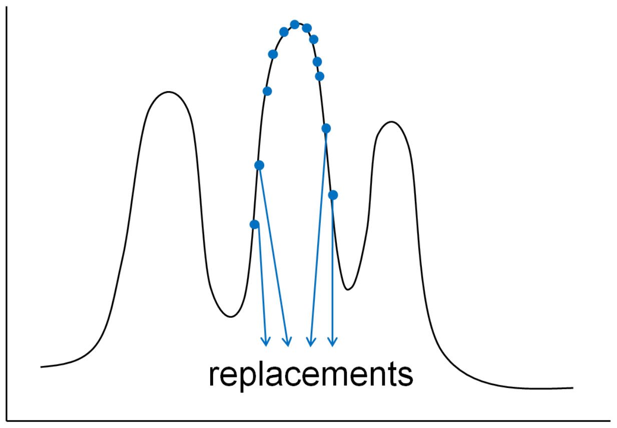


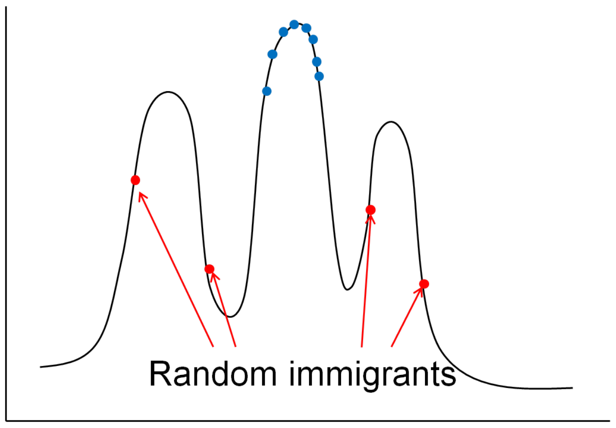
Maintaining Diversity Throughout the Run

Random immigrants – In every generation, the population is partly replaced by randomly generated individuals. Two replacement selection strategies

- replace random individuals,
- replace the worst individuals.

Contrary to hypermutation, it introduces diversity without disrupting the ongoing search process.



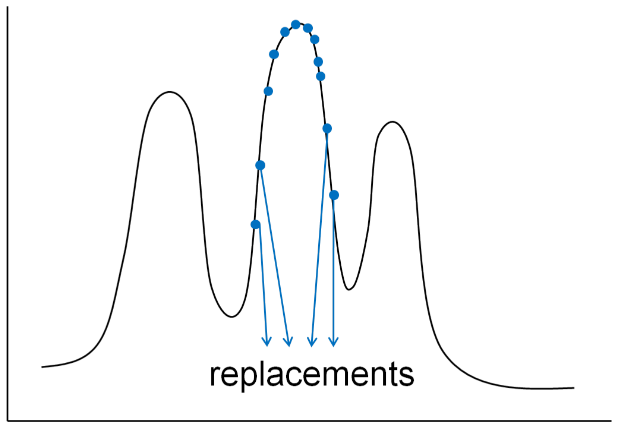


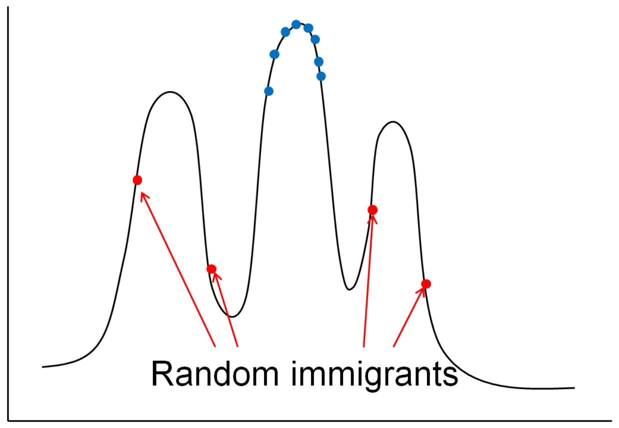
Maintaining Diversity Throughout the Run

Random immigrants – In every generation, the population is partly replaced by randomly generated individuals. Two replacement selection strategies

- replace random individuals,
- replace the worst individuals.

Contrary to hypermutation, it introduces diversity without disrupting the ongoing search process.





Genotypic and phenotypic sharing (niching)

- Tries to spread out the population over multiple peaks.
- Sharing remarkably enhances the GA's ability to track optima in slowly changing environments.

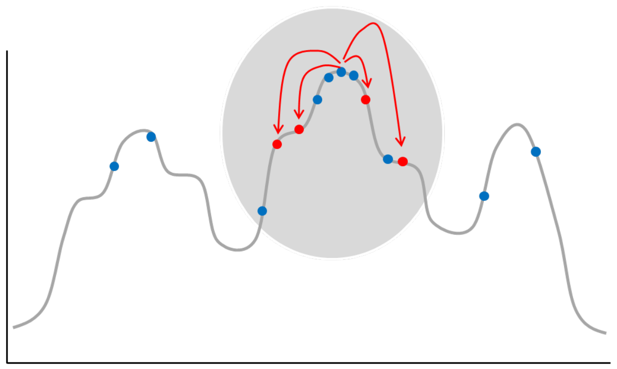
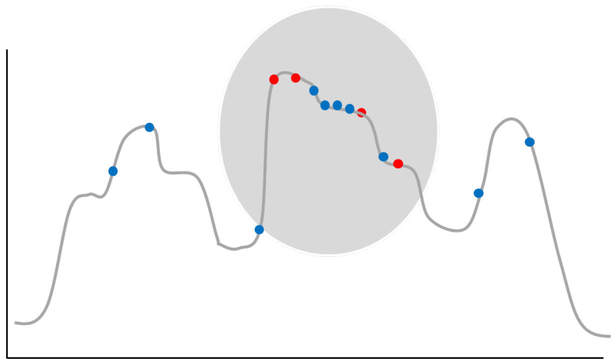
Elitism-Based Immigrants Scheme

Motivation

- In slowly changing environment, random immigrants may may divert the searching process of EAs during each environment and hence may degrade the performance.
- In slowly changing environment, random immigrants may not have any actual effect even when a change occurs because highly-fit individuals in the previous environment may still be quite fit in the new environment.

Elitism-based immigrants scheme – the elite from the previous population is used to guide the immigrants toward the current environment.

- The immigrants are created from the elite of the previous generation with a standard mutation.
- The effect of the elite is limited by the ratio of immigrants – r_i is the portion of the population to be replaced.

Memory-based Approaches: Implicit Memory

Idea – when a change of the environment is detected the search is restarted using information about the search space gathered in the past and stored in a memory.

Especially beneficial in periodically changing environments with repeated occurrences of a small set of environments.

Implicit memory – redundant representations

- Multiploidy (diploidy), with dominance change mechanism – genes specifying one trait are added resulting in value A and the particular trait is determined using a dominance scheme.

  Dominance scheme based on thresholds (reverses after the change):

  - if($A > b_1$) then the trait is 1;
  - if($A < b_2$) then the trait is 0;
  - otherwise the trait is determined at random.

- May slow down convergence and favor diversity.
- Performs comparably to a simple haploid GA with a hypermutation.
- Diploid approach is able to learn two solutions and switch between them almost instantaneously.

  If more than two targets are used, the approach fails.

Memory-based Approaches: Explicit Memory

Explicit memory – specific information is stored and reintroduced into the population at later stages.

- Solutions in the memory that fit the new environment are activated and may help adapt the EA to the new environment more directly than random immigrants.
- Might mislead evolution and prevent it from exploring new regions and discovering new peaks.

What to be stored in the memory

- **Direct memory scheme** – interesting solutions that have been found through the optimization process are stored and reused directly when the environment changes.
- **Associative memory scheme** – stores the solutions together with some environmental information captured at the time the solution is added into the memory.

  The environmental information can be

  - the model from which the solution to be stored was sampled in case of estimation of distribution algorithms,
  - environmental parameters for which the solution was observed (for example, robotic applications).

Memory-based Approaches: Explicit Memory

- **Which individuals should be stored in the memory?**
  - Above average fitness.
  - Not too old.
  - Well distributed across several promising areas of the search space.
- **Memory updating strategy** – selects one memory point to be replaced by the best solution from the population or to be moved toward it.

  Replacement strategies:
  - Importance value – linear combination of the individual's fitness, age and its contribution to diversity.
  - Delete the individual for which we have the maximum variance in the population. The variance for kth individual is calculated as the sum of variances of the alleles over the remaining individuals in the memory.
  - Crowding strategy – the new individual replaces the most similar old individual.

    Alternative: Remove the least fit individual out of the two closest old individuals.
- **Memory retrieval** – use the best individual(s) in the memory to replace the worst individual(s) in the population.

  Can be done periodically or only when the environment changes.

Memory-based Approaches: PBIL with Associative Memory

Population-Based Incremental Learning (PBIL) is a combination of evolutionary optimization using binary representation and competitive learning.

PBIL aims to evolve a real-valued probability vector $\overrightarrow{P} = \{P_1, P_2, \ldots, P_l\}$, where l is the solution length, from which high-quality solutions can be generated with high probability.

PBIL starts from so called *central probability vector* that has a value of 0.5 for each bit of $\overrightarrow{P}$.

```
0 $t = 0$, initialize probability vector $\overrightarrow{P}(0) = \overrightarrow{0.5}$
1 generate a set $S(0)$ of $n$ samples by $\overrightarrow{P}(0)$
2 repeat
3    evaluate samples in $S(t)$
4    learn $\overrightarrow{P}(t)$ toward best sample $\overrightarrow{B}(t)$ in $S(t)$ by
5              $P_i(t+1) = (1-\alpha) * P_i(t) + \alpha * B_i(t)$, $i = \{1, \ldots, l\}$
6    mutate $\overrightarrow{P}(t)$ toward *central probability vector*
7    $t = t + 1$
8    generate a set $S(t)$ of $n$ samples by $\overrightarrow{P}(t)$
9 until(termination condition)
```

Memory-based Approaches: PBIL with Associative Memory

Explicit associative memory stores couples composed of **the best sample** in $S(t)$ and its **probability vector** $\overrightarrow{P}(0)$ (*the current environmental information*).

Memory update

- Memory member with its sample $\overrightarrow{B}_M(t)$ closest to the best population sample $\overrightarrow{B}(t)$ in terms of the Hamming distance is found.
- If $\overrightarrow{B}(t)$ is better than $\overrightarrow{B}_M(t)$, it is replaced by $\overrightarrow{B}(t)$. The probability vector $\overrightarrow{P}(t)$ is updated as well. Otherwise, the memory remains unchanged.

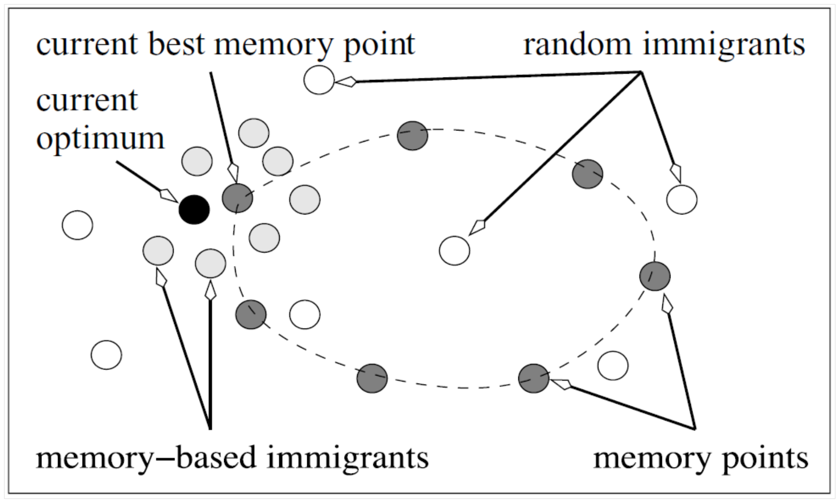
Memory retrieval – the memory is reevaluated every iteration. If any memory sample has its fitness changed, the environment is detected and the following action is taken

- The memory probability vector associated with the best memory sample, $\overrightarrow{P}_{B_M}(t)$, will replace the current probability vector $\overrightarrow{P}(t)$ if $\overrightarrow{B}_M(t)$ is better than $\overrightarrow{B}(t)$. Otherwise, the probability vector $\overrightarrow{P}(t)$ remains unchanged.

Memory-Based Immigrants Scheme

Combination of random immigrants and memory-schemes – the memory is used to guide the immigrants to make them more biased to the current environment than random immigrants.

- For every generation the memory is re-evaluated and the best individual is retrieved as the base to create immigrants.
- $r_i \times n$ immigrants, where n is the size of the population, are generated by mutation and replace the worst individuals in the population.



- Random immigrants are distributed over the whole search space.
- Memory-based immigrants are distributed around the base memory solution.

Maintaining Diversity Using Redundant Representation: GARB

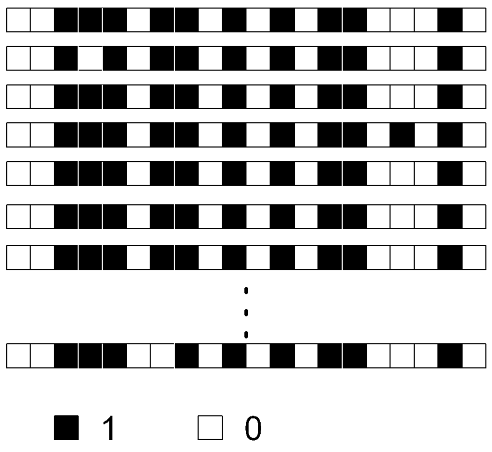
GA with Real-coded Binary Representation

Motivation

- using redundant representation, where many different genotypes map to the same phenotype, would increase the explorative power of the EA and decrease the probability of getting stuck in a local optimum.

Realization

- real-coded (redundant) binary representation.



Effect

- population can not converge to the homogeneous state so that the premature convergence can not take place.

GARB: Representation

Pseudo-binary representation

- binary gene values coded by real numbers from the interval $\langle 0.0, 1.0\rangle$.

$$\begin{aligned} interpretation &= \text{1} \quad \text{, for } r \geq 0.5 \\ &= \text{0} \quad \text{, for } r < 0.5 \end{aligned}$$

Example: $ch_1 = [0.92, 0.07, 0.23, 0.62]$, $ch_2 = [0.65, 0.19, 0.41, 0.86]$

$$interpretation(ch_1) = interpretation(ch_2) = [1001]$$

Gene strength – gene's stability measure

- The closer the real value is to 0.5 the weaker the gene is.
  "one-valued genes": $0.92 >_S 0.86 >_S 0.65 >_S 0.52$
  "zero-valued genes": $0.07 >_S 0.19 >_S 0.35 >_S 0.49$
  where "$>_S$" is the "*stronger relation*".

GARB: Gene-Strength Adaptation

Vector P stores the frequencies of ones at every position in the current population.

- Example: $P = [0.82, 0.17, 0.35, 0.68]$
  - 82% of ones at the first position,
  - 17% of ones at the first position,
  - 35% of ones at the first position,
  - 68% of ones at the first position,

Every offspring gene is adjusted depending on

- its interpretation,
- the relative frequency of ones at given position in the population.

GARB: Gene-Strength Adaptation cnd.

Adjustment

- Weakening – a gene at the ith position is weakened proportionally to $P[i]$ if it interprets as the binary value that is more frequently sampled at the ith position in the current population.

  The gene's value is shifted towards 0.5
- Strengthening – A gene at the ith position is strengthened proportionally to $P[i]$ if it represents the binary value that is less frequently sampled at the ith position.

  The gene's value is shifted towards the corresponding extreme 0.0 or 1.0.

Realization

- Strengthening binary one genes, weakening binary zero genes (gene at ith position):

$$gene'_i = gene_i + c * (1.0 - P[i])$$

- Strengthening binary zero genes, weakening binary one genes (gene at ith position):

$$gene'_i = gene_i - c * P[i]$$

  where c stands for a maximal gene-adaptation step: $c \in (0.0, 0.2)$

GARB: Gene-Strength Adaptation cnd.

Effect

- If some allele begins to prevail in the population,
  1. the corresponding genes are weakened in subsequent generations,
  2. at some point the genes might change their interpretation (shift over the threshold 0.5) and the frequency of the allele decreases.
- Frequency of a given allele is controlled by contradictory pressures
  - the convergence to the optimal solution pressure,
  - the population diversity preservation pressure.

GARB: Boosting-up the Exploitation

Genotype of promising solutions should be stabilized for subsequent generations in order to disable rapid changes in their genotype interpretation.

Promising solutions are newly generated solutions that are better than their parents

- all genes are re-scaled (strengthened) - zero-valued genes are set to be close to 0.0 and one-valued genes are set to be close to 1.0.

  Example:

$$\begin{aligned} ch &= (0.71, 0.45, 0.18, 0.57) \\ & \qquad\qquad\quad \downarrow \\ ch' &= (0.97, 0.03, 0.02, 0.99) \end{aligned}$$

Effect – genes survive with unchanged interpretation through more generations.

GARB: Algorithm

Standard generational genetic algorithm + diversity preservation related features

```
begin
initialize(OldPop)
repeat
    calculate P[] from OldPop
    repeat
        select Parents from OldPop
        generate Children
        adjust Children genes
        evaluate Children
        if Child is better than Parents
            then rescale Child
        insert Children to NewPop
    until NewPop is completed
    switch OldPop and NewPop
until termination condition
end
```

GARB: Moving Optimum Problem

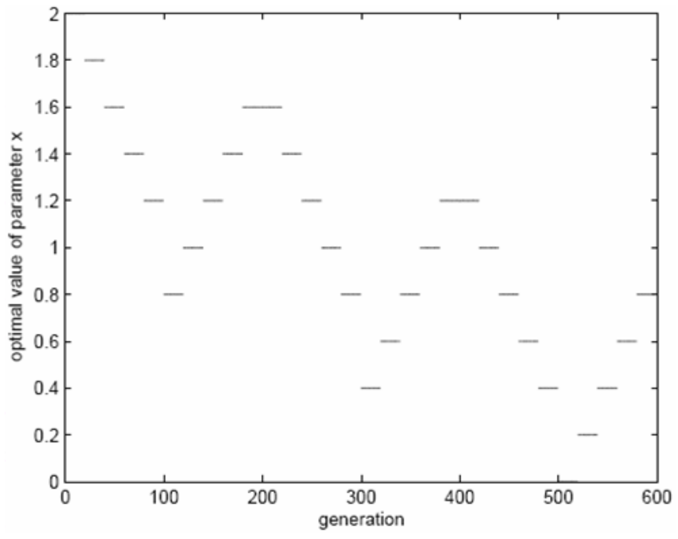
Ošmera's dynamic problem:

$$g_1(x,t) = 1 - e^{-200(x-c(t))^2}$$

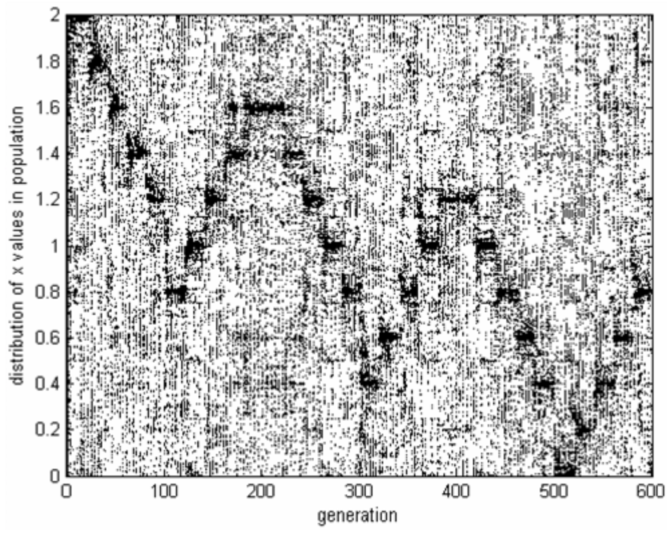
where x is represented by a bit-string of length 31 and normalized to give a value in the range $(0.0, 2.0)$, $t \in \{1, 1000\}$ is the time-step and

$$c(t) = 0.2 \cdot (|\lfloor t/100 \rfloor - 5| + |5 - (\lfloor t/20 \rfloor mod 10)|)$$

specifies the changes in the optimal bit-string.



Moving optimum

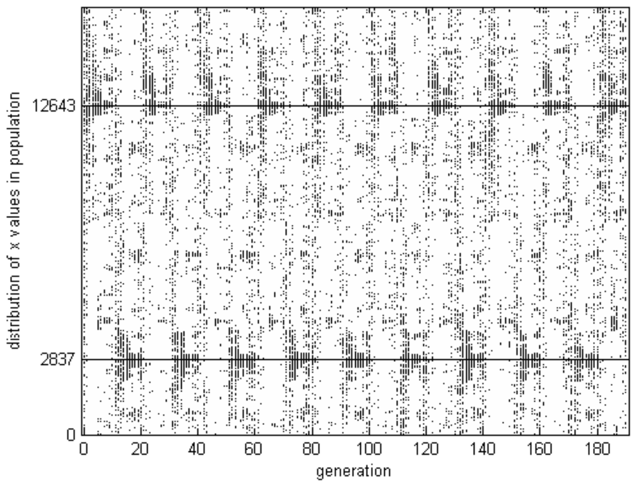


Evolution of the population diversity

GARB: Oscillating Knapsack Problem

Oscillating Knapsack Problem (String matching problem)

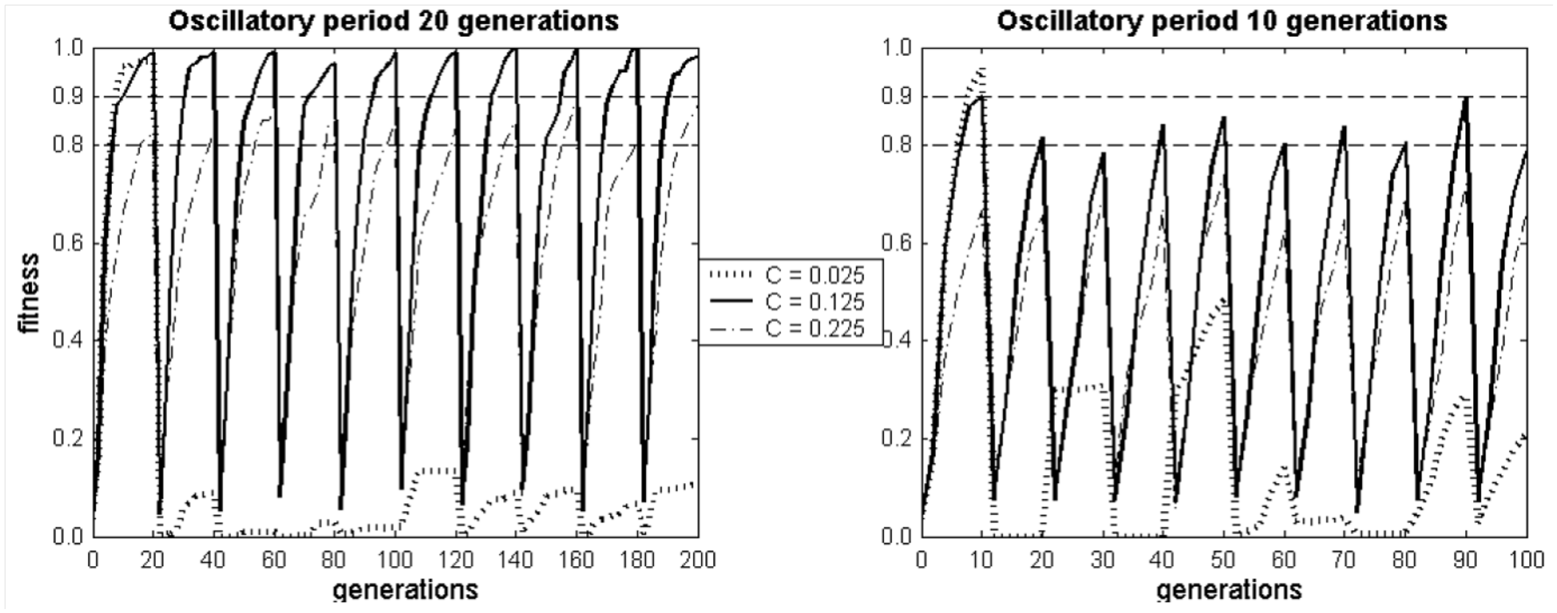
- Goal is to fill a knapsack with objects from an available set of size n, such that the sum of object weights is as close as possible to the target weight t.
- 14 objects, each of them has weight $w_i = 2^i$, where $i = 0 \dots 13$.
- Optimum fitness is 1.0 and decreases towards 0.0 as the Hamming distance to the optimum solution string increases.
- Target weight oscillates between values 12643 and 2837 (9 bits difference).



Evolution of the population diversity

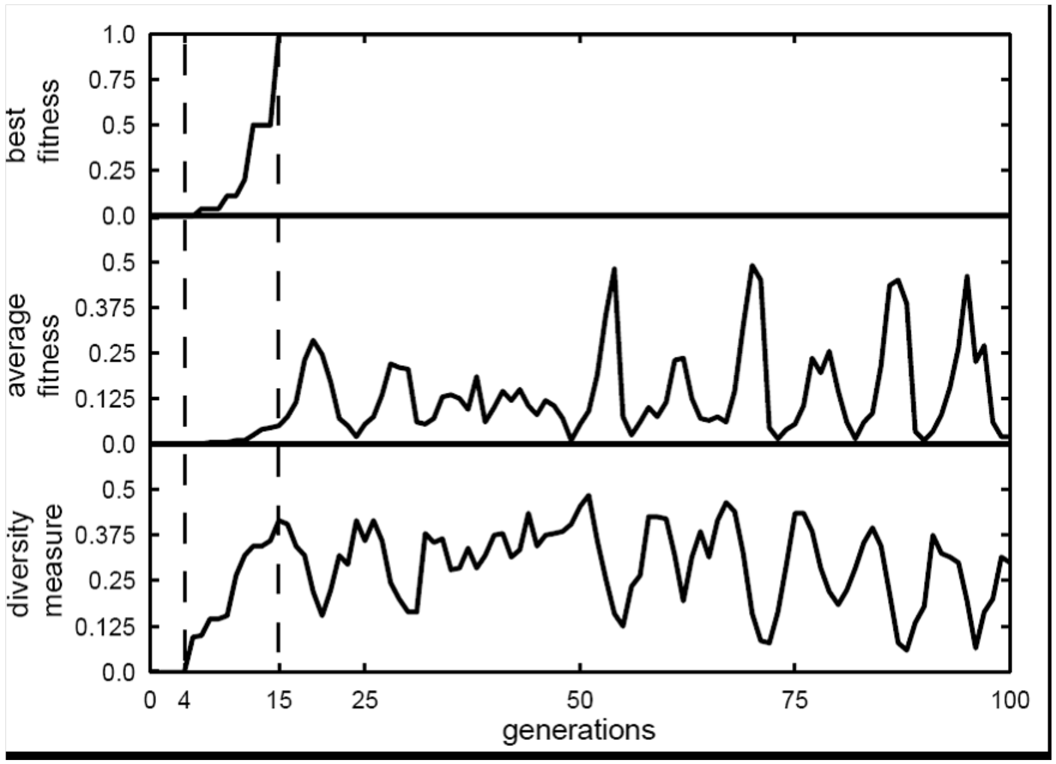
GARB: Oscillating Knapsack Problem cnd.

Average performance of GARB



GARB: Recovery from Completely Homogeneous State

GARB exhibits abilities to recover even from a completely homogeneous state, where the population consists of multiple copies of the same solution (the wrong target in this case).



GARB: Summary

- **Redundant representation** realized by real-valued genes interpreted as binary 0/1 values.
- **Gene-strength adjustment** plays a role of a *directed self-adaptive mutation* – effect on every gene is directly proportional to the population convergence rate at the given position.
- Strong **capabilities to prevent a ”premature” convergence** that makes it suitable for optimizations in dynamically changing environments.

Concluding Remarks

- Standard EAs designed for stationary optimization problems suffer from so called *convergence problem* when used for solving dynamic optimizations, i.e. once the EA converges to an optimum its population gets homogeneous and the EA looses its ability to adapt to a change in the environment.
- Explicit memory and redundant multiploid coding are well-suited for cases in which the optimum oscillates periodically. Multiploid coding approach seems to be effective only if the location of optimums are very limited.
- Redundant coding used in GARB is efficient even for continuous and random changing environments.

Reading

- Branke J.: Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems, 1999
  http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.8897
- Branke J.: Evolutionary Approaches to Dynamic Optimization Problems - A Survey, 2001
  `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.4619`
- Yang S.: Genetic Algorithms with Memory- and Elitism-Based Immigrants in Dynamic Environments. Evolutionary Computation, Vol. 16, No. 3, pp. 385-416, 2008.
- ECiDUE: Evolutionary Computation in Dynamic and Uncertain Environments
  http://www.cs.le.ac.uk/people/sy11/ECiDUE/