Evolutionary Algorithms for Dynamic Optimization Problems

Patryk Filipiak

27 Jan 2014

Dynamic Optimization Problems (DOPs)

It is a special class of dynamic problems that are solved ***online*** *by an optimization algorithm* ***as time goes by***.

(Yang, Nguyen, Li, 2013)

Formal definition

Objective function

Let $d > 0$, $D \subseteq \mathbb{R}^d$. Minimize

$$F^{(\alpha)} : D \to \mathbb{R},$$

where α is a vector of parameters changing as time goes by.

Notation:

$$F^{(t)} = F^{(\alpha_t)},$$

where α_t is a vector of parameters α at the time step $t \in \mathbb{N}_+$.

Formal definition

Constraints

Analogously, define the constraints

$$G_i^{(\alpha)} : \mathbb{R}^d \to \mathbb{R}, \quad i = 1, \ldots, m.$$

Notation: $G_i^{(t)} = G_i^{(\alpha_t)}$, $t \in \mathbb{N}_+$.

Aim

For all $t \in \{t_1, t_2, \ldots, t_k\} \subset \mathbb{N}_+$ find $x^{(t)} \in \mathbb{R}^d$ such that

$$x^{(t)} = \arg\min\{F^{(t)}(x) : x \in \mathbb{R}^d \wedge \forall_{i=1,\ldots,m}\ G_i^{(t)}(x) \geqslant 0\}.$$

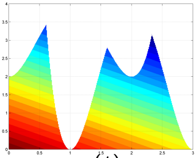
Real-world examples

- real-time optimization of investment portfolio,
- resource management (e.g. machines, rooms, vehicles) with the ability to adapt to changing demands,
- air traffic control.

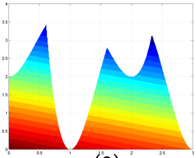
(Bui, Branke, Abbass, 2005)

Sample benchmark

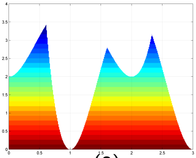
2D plots of the objective function (including constraints)



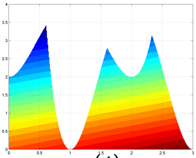
$F^{(1)}$



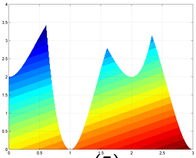
$F^{(2)}$



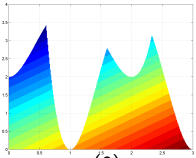
$F^{(3)}$



$F^{(4)}$



$F^{(5)}$



$F^{(6)}$

DOP = iterated Stationary Optimization Problem?

- Discrete time sampling $t \in \mathbb{N}+$ allows for transforming any DOP into iterated Stationary Optimization Problems (SOPs).
- Iterations of such SOPs can be solved using classical EAs.

Re-initialization

Each iteration of SOP begins with re-initialization of a population. As a result:

- Even small environmental changes imply starting new optimization process from scratch.
- Re-initialization results in the information loss between consecutive time steps.

Alternatively, it is possible to start t-th SOP with the individuals being the output of $(t-1)$-th SOP.

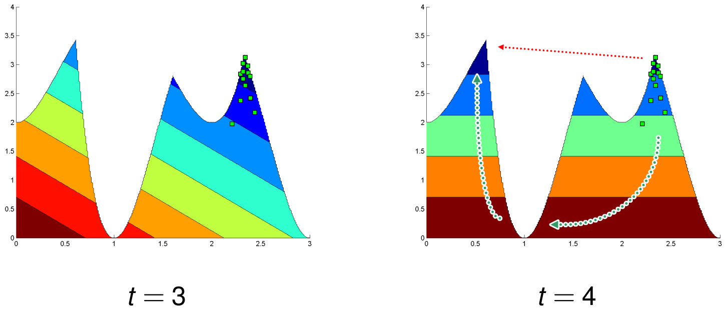
Low reactivity to environmental changes

It takes to much time for a converged population from $(t-1)$-th SOP to adapt to the new state of the environment at t-th SOP.

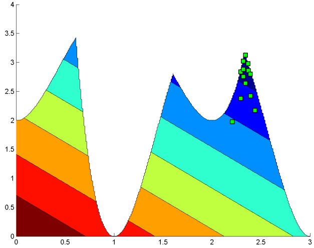
Dynamic constraints

A presence of time-dependent constraint functions $G_i^{(t)}$ imply that at least some of feasible individuals at the time step $t-1$ may become infeasible at the time step t.

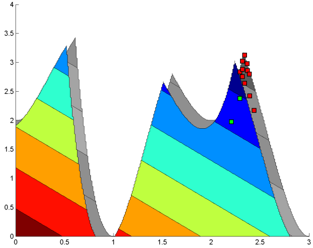
Example – low reactivity to environmental changes



Example – dynamic constraints



$t = 3$



$t = 4$

Continuous adaptation

Continuous adaptation only makes sense when the landscapes before and after the change are sufficiently correlated, otherwise it would be at least as efficient to restart the search from scratch.

(Branke et al., 2000)

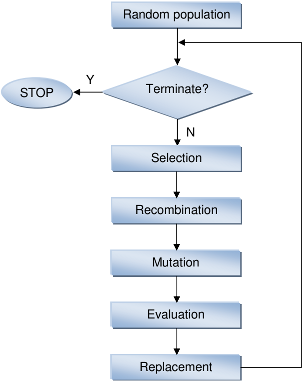
Simple Genetic Algorithm is not an option

Stagnation

Lack of ability to track moving optima

Premature convergence

Lack of ability to localize newly appearing optima



Changes detection required

Continuous adaptation requires information about the exact moments of the environmental changes.

- **Synchronous changes** – fixed time periods between changes (e.g. weekly updates, daily reports),
- **Asynchronous changes** – varying time periods between changes; a change detection mechanism is required:
  - random sampling of the search space,
  - frequent re-evaluation of a population,
  - anticipation of changes (based on the past changes).

Introducing diversity approach

Diversification is introduced as soon as the environmental change is detected.

while not Termination condition **do**
  Perform $k > 0$ iterations of a classic algorithm.
  Look for changes in the environment.
  if Change is detected **then**
    Diversify a significant fraction of population.
  end if
end while

Introducing diversity approach

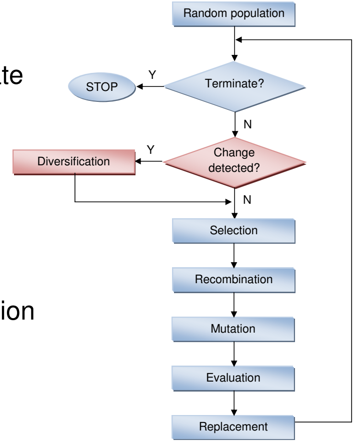
Triggered Hypermutation GA

Triggered increase of mutation rate

Drawbacks

- based on changes detection mechanism,
- fixed rate of hypermutation,
- fixed duration of hypermutation period.

(Cobb, 1990)



Maintaining diversity approach

Population is diversified in each generation by

- removing random individuals,
- removing individuals with lowest fitness,
- removing *similar* individuals,
- promoting *distant* individuals.

Removed individuals are often replaced with *random immigrants*.

Maintaining diversity approach

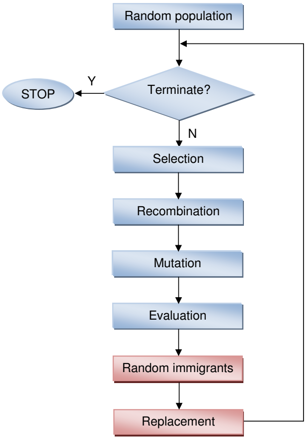
Random Immigrants GA (RIGA)

Iterated introduction of random immigrants

Drawbacks

- size of immigrants fraction needs to be estimated,
- dozens of very weak individuals among immigrants.

(Grefenstette, 1992)



Memory based approach

Population is extended with a buffer of the former best individuals.

- *explicit* memory – full copies of individuals,
- *implicit* memory – metadata about individuals, e.g.
  - averaged or discrete forms of chromosomes,
  - probability distributions.

Memory based approach

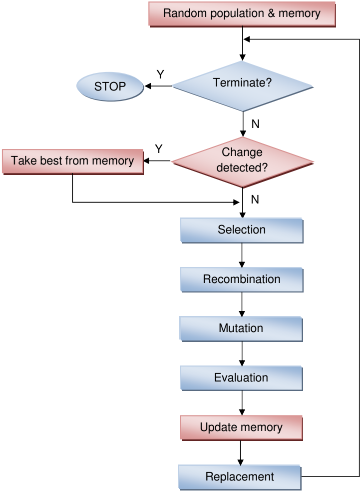
Memory Enhanced GA (MEGA)

Buffer for the former best individuals

Drawbacks

- size of buffer needs to be estimated,
- applicability to *recurrent* problems only.

(Yang, 2005)



CHC algorithm

C*ross-generational elitist selection,*
H*eterogeneous recombination, and*
C*ataclysmic mutation*

In other words:

(C) best individual remains unmodified.

(H) only “sufficiently distant” individuals are mated (*incest avoidance*).

(C) mutation of nearly all individuals.

(Eshelman, 1991)

CHC algorithm – selection

Hamming distance (for chromosomes)

Number of unmatched *allels* in the two chromosomes.

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}, \quad d_H = 4.$$

Selection mechanism

- All individuals are mated randomly.
- Only those pairs of individuals with Hamming distance exceeding the threshold value $d > 0$ are allowed to recombine.

CHC algorithm – recombination (crossover)

- First, a half of unmatched allels is picked randomly

$$\begin{pmatrix} 1 & \underline{\mathbf{0}} & 0 & 1 & 1 & 1 & 0 & \underline{\mathbf{1}} \\ 1 & \underline{\mathbf{1}} & 0 & 1 & 0 & 1 & 1 & \underline{\mathbf{0}} \end{pmatrix}$$

- Then, the selected allels are exchanged

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

- Note that the above process preserves the Hamming distance of parents.

CHC algorithm – re-initialization

Re-initialization plays the role of the mutation:

- Only one best individual remains unmodified.
- The rest of individuals are replaced with clones of the best individual with $r > 0$ randomly perturbed allels.

Let $L > 0$ be the length of chromosome:

- r near 0 – fast convergence, possibility of getting trapped into local optima;
- r near L – slow convergence, large diversity of individuals;

CHC pseudo-code

Parameters:

- d – threshold for Hamming distance,
- L – chromosome length.
- r – number of allels mutated during re-initialization.

$i = 0$; $d = L/4$; Initialize(P_0)
while not TerminationCondition(P_i) **do**
  Evaluate(P_i)
  P'_i = Selection(P_i)
  if $P'_i \neq \emptyset$ **then**
    C_i = Crossover(P'_i)
    Evaluate(C_i)
    P_{i+1} = Re-initialize$(P_i \cup C_i, r)$
    $d = L/4$
  else
    $d = d - 1$
  end if
  $i = i + 1$
end while

CHC algorithm

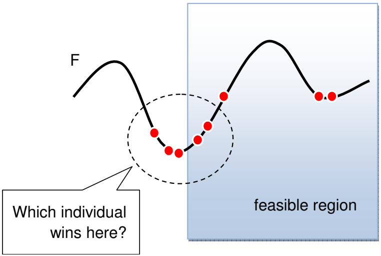
Advantages

- rapidness,
- high reactivity to the environmental changes,

Disadvantages

- loss of information due to the re-initialization,
- incapability of dealing with multimodal problems.

Constraints handling



- Should infeasible individuals be removed or repaired?
- What is the fitness of an infeasible individual?
- How to compare two infeasible individuals?

IDEA (Infeasibility-Driven Evolutionary Algorithm)

- Maintaining a fixed-sized fraction of “good yet infeasible” individuals.
- Rank method used in IDEA:
  - All feasible individuals are ranked 0.
  - The remaining (i.e. infeasible) individuals are ordered according to their *violation measure* then ranked 1, 2, 3, . . .

(Singh, Isaacs, Ray, Smith, 2008)

IDEA(N, α)

Parameters: N – population size, $0 < \alpha < 1$ rate of infeasible fraction.

$N_{inf} = \alpha \cdot N$
$N_f = N - N_{inf}$
$P_1 = \text{InitPopulation}()$
$\text{Evaluate}(P_1)$
while not $\text{TerminationCondition}(P_i)$ **do**
  $C_i = \text{Crossover}(P_i)$
  $C_i = \text{Mutation}(C_i)$
  $\text{Evaluate}(C_i)$
  $(S_f, S_{inf}) = Split(P_i + C_i)$
  $Rank(S_f)$
  $Rank(S_{inf})$
  $P_{i+1} = S_f(1 : N_f) + S_{inf}(1 : N_{inf})$
end while

IDEA adapted to DOPs

Although IDEA was designed for SOPs, it is also easily applicable for DOPs.

Singh et al. (2009) proposed the simple modification for that purpose, by introducing Sub-evolve step.

- Sub-evolve step is basically the original IDEA.
- The main loop verifies if the environment has changed since the last time step. If so, it re-evaluates the whole population.

IDEA adapted to DOPs

N_G – number of generations

$P1 = \text{Initialize}()$
Evaluate(P_1)
for $i = 2$ to N_G **do**
  if the function has changed **then**
    Evaluate(P_{i-1})
  end if
  $C_{i-1} = \text{Sub-evolve}(P_{i-1})$
  Evaluate(C_{i-1})
  $P_i = \text{Reduce}(P_{i-1} + C_{i-1})$
end for

IDEA adapted to DOPs

Advantages

- good performance in constrained optimization problems,
- applicability to multi-objective problems.

Disadvantages

- necessity to verify feasibility of the individuals during each generation,
- definition of violation measure dedicated to the examined problem.

Summary

- DOPs model many real-world optimization problems.
- DOPs are solved with the dedicated adaptations of classical EAs.
- There is no “the one good EA” for all DOPs. There are few classes of EAs instead, each of which suits better a given class of DOPs.
- Developing EAs for DOPs is among the current research problems.