Evolutionary Continuous Dynamic Optimization

Danial Yazdani[1] and **Xin Yao**[1,2]

[1]Department of Computer Science and Engineering,

[2] Research Institute of Trustworthy Autonomous Systems (RITAS),

Southern University of Science and Technology, Shenzhen, China

IEEE SSCI 2021

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Outline

- Introduction: Dynamic Optimization Problems
- Evolutionary Dynamic Optimization
- Benchmark Problems
- Performance Analysis Methods
- Future Research Directions

- Introduction: Dynamic Optimization Problems
- Evolutionary Dynamic Optimization
- Benchmark Problems
- Performance Analysis Methods
- Future Research Directions

Dynamic Optimization Problems (DOPs)

- Change is an unavoidable part of many optimization problems and adaptation is necessary to tackle them.
- A DOP can be defined as:

$$\begin{aligned} \max\ & f\left(\vec{x},\vec{\alpha}^{(t)}\right), \vec{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d \\ s.t.:\ & g(\vec{x}) \leq 0 \\ & h(\vec{x}) = 0 \end{aligned}$$

  - Where f is the objective function, $\vec{x}$ is a solution in the search space, $\vec{\alpha}$ s a vector of time-varying control parameters, and $t \in [1, T]$ is the time index.
- Can be converted to unconstrained optimization using penalty method, Lagrangian, etc.
- Our focus is unconstrained DOPs whose environmental changes happen in discrete time steps:

$$\left\{f\left(\vec{x},\vec{\alpha}^{(t)}\right)\right\}_{t=1}^{T} = \left\{f\left(\vec{x},\vec{\alpha}^{(1)}\right), f\left(\vec{x},\vec{\alpha}^{(2)}\right), \cdots, f\left(\vec{x},\vec{\alpha}^{(T)}\right)\right\}$$

Dynamic Optimization Problems (DOPs)

$$\left\{f\left(\vec{x},\vec{\alpha}^{(t)}\right)\right\}_{t=1}^{T}=\left\{f\left(\vec{x},\vec{\alpha}^{(1)}\right),f\left(\vec{x},\vec{\alpha}^{(2)}\right),\cdots,f\left(\vec{x},\vec{\alpha}^{(T)}\right)\right\}$$

- In the *t*th environment, $f\left(\vec{x},\vec{\alpha}^{(t)}\right)=f^{(t)}(\vec{x})$ is used as the fitness function.
- After each environmental change, the search space alters.
- One common goal is to find the global optimum in each environment.
- It is commonly assumed in the literature that there is a degree of similarity between the successive environments.
  - In such DOPs, the knowledge obtained from previous environments can be useful for accelerating the optimization process in the current environment.
- If the environmental changes are very severe and there is no similarity between successive environments, the optimization method must be reinitialized after each environmental change.

Dynamic Optimization Problems (DOPs): Classification

De Jong [1]

- DOPs With Drifting Landscapes
    - Change severity of the environment is very gradual, but the change frequency is high.
- DOPs With Significant Morphological Changes
    - Fitness of different regions can either increase or decrease after environmental changes.
- Periodical DOPs
    - Environmental changes in these DOPs show reappearing/cyclic patterns where the landscape visits a finite set of states repeatedly.
- DOPs With Abrupt Changes
    - Huge change severity and very low similarity between successive environments.

[1] K. De Jong, "Evolving in a changing world," in *Proc. Int. Symp. Methodol. Intell. Syst.*, 1999, pp. 512–519.

Dynamic Optimization Problems (DOPs): Classification

Besides the change severity and frequency, Branke and Schmeck [1] also used the following criteria:

- Predictability
  - Some aspects of the problem follow a regular pattern, so they can be learned and then predicted.
- Change visibility
  - In DOPs with visible changes, the EDOs are informed about the occurrence of environmental changes.
- Aspects of changes
  - Indicates which parts of the problem, such as dimension, objective function, and/or constraints, change over time.

[1] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Eds. Berlin, Germany: Springer, 2003, pp. 239–262.

Dynamic Optimization Problems (DOPs): Classification

Other criteria:

- Policy of deploying solution
  - Tracking the moving optimum (TMO) [1]: the main goal is to find a desirable solution (i.e., the global optimum) for deployment in each environment.
  - Robust optimization over time (ROOT) [2]: the main goal is to find solutions for deployment, which are not necessarily the best in term of fitness, but their quality over longer time horizons involving multiple environmental changes remain acceptable.

- Time linkage [3]: deployed solution in one environment influences the environment(s) encountered in the future.

[1] J. Branke, “Memory enhanced evolutionary algorithms for changing optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 3. Washington, DC, USA, 1999, pp. 1875–1882.

[2] X. Yu, Y. Jin, K. Tang, and X. Yao, “Robust optimization over time— A new perspective on dynamic optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 1–6.

[3] T. T. Nguyen and X. Yao, “Dynamic time-linkage problems revisited,” in *Proc. Workshops Appl. Evol. Comput.*, 2009, pp. 735–744.

Dynamic Optimization Problems (DOPs): Classification

- These classifications consider different criteria for categorizing DOPs.
- What can we learn from these classifications? How can we use them when we are designing an algorithm? Here are some examples:
  - To solve *DOPs with abrupt changes*, the obtained knowledge from previous environments is considerably less useful. Reinitializing the optimization method after each environmental change might be the best option.
  - To solve *DOPs with significant morphological changes*, we need a high global diversity (e.g., using multi-population methods) to make the algorithm capable of locating global optimum after changes.
  - To solve *periodical DOPs*, we should use an explicit archive to store some gathered information from each environment and use them when these environments reappear in the future.
  - To solve *DOPs with visible environmental changes*, the optimization algorithm does not need to use a change detection component since the algorithm is informed about occurrence of each change.
  - To solve a *ROOT*, we also need to consider the future fitness and robustness of solutions.

- Introduction
- **Evolutionary Dynamic Optimization**
- Benchmark Problems
- Performance Analysis Methods
- Future Research Directions

Evolutionary Dynamic Optimization

- Methods for solving DOPs [1]
  - Heuristics
  - Dynamic Programming
  - Stochastic Programming
  - Metaheuristics
- Evolutionary algorithms (EA) and swarm intelligence (SI) methods have been vastly used for optimizing DOPs.

[1] T. T. Nguyen, “Continuous dynamic optimisation using evolutionary algorithms,” Ph.D. dissertation, Dept. Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2011.

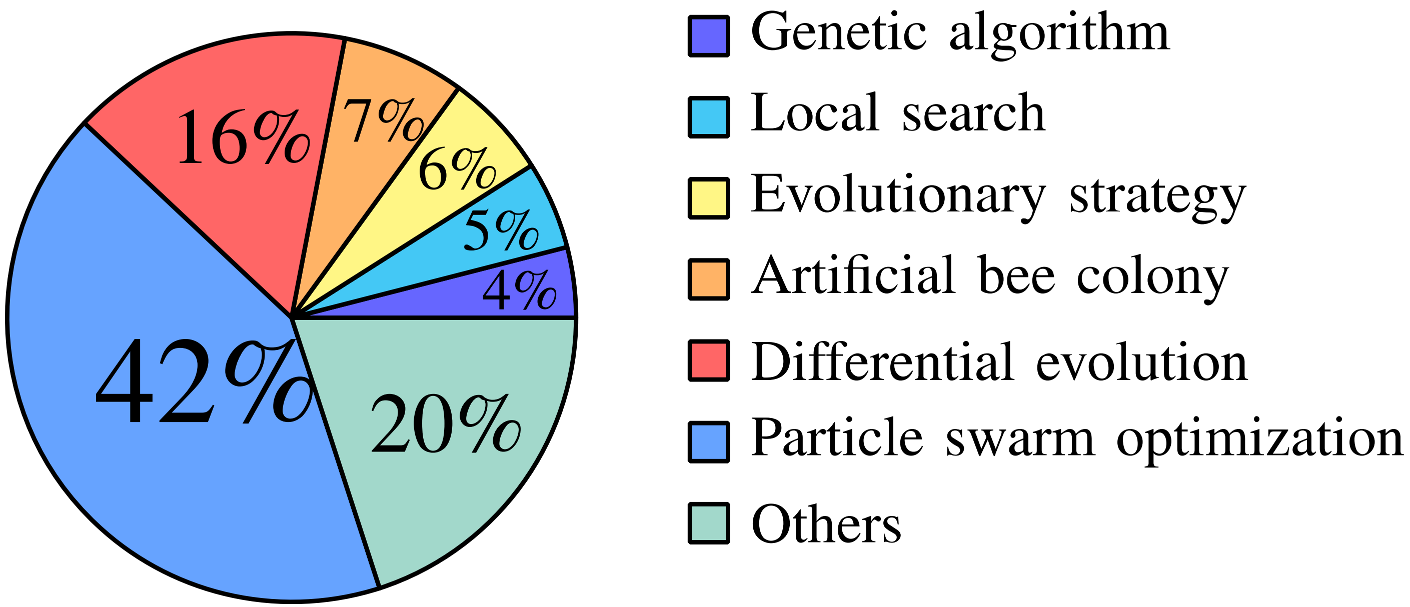
Evolutionary Dynamic Optimization

Challenges of DOPs:

- Global diversity loss
  - This occurs due to the intrinsic nature of EAs and SIs converging to promising regions.
- Local diversity loss
  - This occurs since individuals usually collapse to an optimum during exploitation.
- Limited computational resources
  - Usually, there are limited available computational resources during each environment.
- Outdated memory
  - After each environmental change, any stored fitness values, which were calculated based on the previous environment, will become outdated.

Evolutionary Dynamic Optimization

- Usage percentages of different EA/SI methods in the field [1].



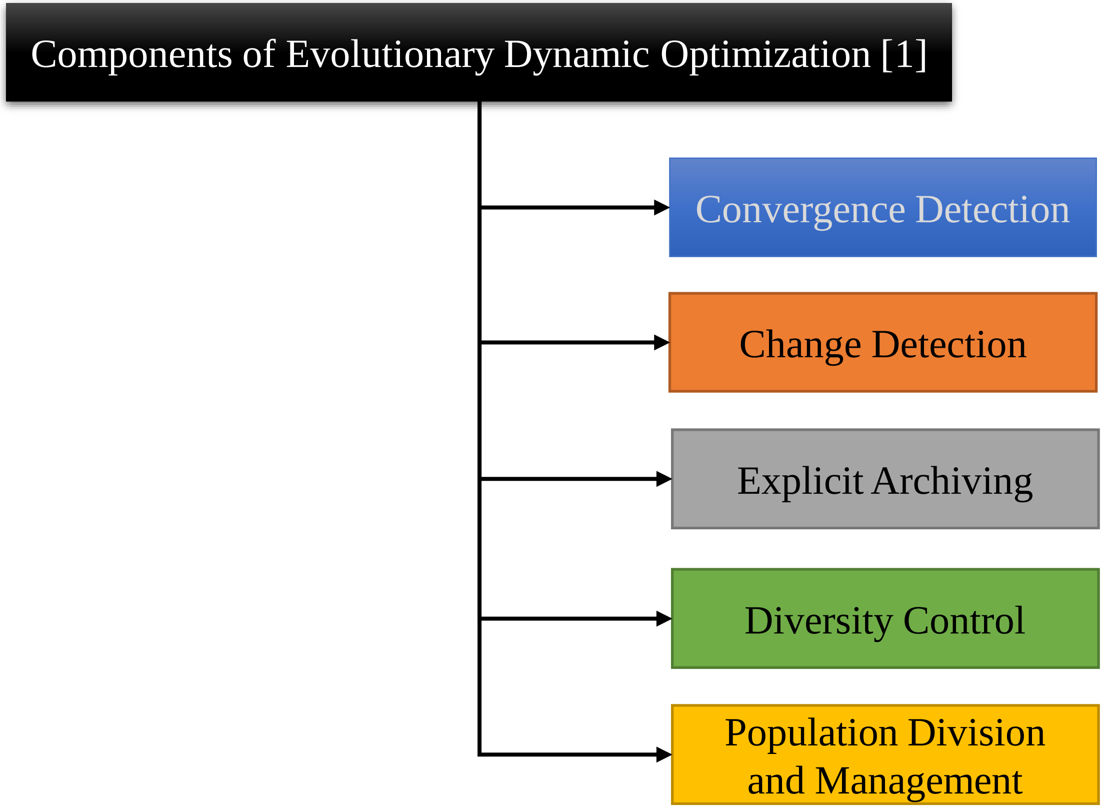
[1] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A survey of evolutionary continuous dynamic optimization over two decades—Part B,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 630–650, Aug. 2021.

Evolutionary Dynamic Optimization: Overall Picture

EA/SI methods are originally designed for solving static optimization problems. To solve DOPs, they are usually used together with some other components:

- In some DOPs, the optimization algorithm needs to detect environmental changes. This is very important because the algorithm needs to react to changes and prepare for the new environment.
- After convergence, the population loses its diversity. Consequently, after an environmental change, both exploration and exploitation capabilities of the population are hindered. EDOs usually use some diversity control components to address this issue.
- Using obtained knowledge from previous environments is beneficial. To this end, some algorithms use an explicit archive to store some obtained information.
- To increase the exploration ability and also the capability of monitoring/covering multiple promising regions, EDOs usually use multiple sub-populations. To manage several sub-populations, these EDOs use some population division and management components.
- Many procedures of diversity control, explicit archiving, and population management components are triggered after convergence of a sub-population. Therefore, EDOs usually also use a convergence detection component.

Evolutionary Dynamic Optimization: Components



Components of Evolutionary Dynamic Optimization [1]

Convergence Detection

Change Detection

Explicit Archiving

Diversity Control

Population Division and Management

- Usually, the procedure of EA/SI is independent of these components.
- In each iteration of an EDO, an internal iteration of EA/SI is run for each sub-population chosen by the population division and management component.
- Some of these components run every iteration.
- Some of these components are triggered when some certain conditions are met, such as after environmental changes or after convergence.

[1] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A survey of evolutionary continuous dynamic optimization over two decades—Part A,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 609 - 629, Aug. 2021.

Convergence Detection

- Convergence detection by monitoring the fitness of the best found position by the sub-population.

$$\begin{cases} \text{Converged} & f^{(t)}\left(\vec{\boldsymbol{g}}^{*(i)}\right) = f^{(t)}\left(\vec{\boldsymbol{g}}^{*(i-k)}\right) \\ \text{Unconverged} & f^{(t)}\left(\vec{\boldsymbol{g}}^{*(i)}\right) > f^{(t)}\left(\vec{\boldsymbol{g}}^{*(i-k)}\right) \end{cases} \quad [1]$$

Where $\vec{\boldsymbol{g}}^{*(i)}$ is the best found position by the sub-population in the ith iteration.

- This method only can detect convergence if the best found position has not improved during the last k iterations. This shortcoming is solved in [2]:

$$\begin{cases} \text{Converged} & f^{(t)}\left(\vec{\boldsymbol{g}}^{*(i)}\right) - f^{(t)}\left(\vec{\boldsymbol{g}}^{*(i-k)}\right) \le \epsilon \\ \text{Unconverged} & f^{(t)}\left(\vec{\boldsymbol{g}}^{*(i)}\right) - f^{(t)}\left(\vec{\boldsymbol{g}}^{*(i-k)}\right) > \epsilon \end{cases}$$

Where ϵ is a small constant.

[1] M. C. du Plessis and A. P. Engelbrecht, “Differential evolution for dynamic environments with unknown numbers of optima,” *J. Global Optim.*, vol. 55, no. 1, pp. 73–99, 2013.
[2] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, “A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization,” *Appl. Soft Comput.*, vol. 13, no. 4, pp. 2144–2158, 2013.

Convergence Detection

- Convergence detection by monitoring the spatial size of a sub-population
  - A sub-population *a* has converged if its spatial size s_a is smaller than a threshold.

$$s_a = \max_{i \in a} \| \vec{\boldsymbol{g}}^* - \vec{\boldsymbol{x}}_i \| \qquad [1]$$

Where $\vec{\boldsymbol{g}}^*$ is the best found position by the *a*th sub-population and $\vec{\boldsymbol{x}}_i$ is the position of the *i*th individual in the *a*th sub-population.

$$s_a = \max_{i,j \in a} \left\| \vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j \right\| \qquad [2]$$

$$s_a = \frac{1}{n_a} \sum_{i \in a} \| \vec{\boldsymbol{c}}_a - \vec{\boldsymbol{x}}_i \| \qquad [3]$$

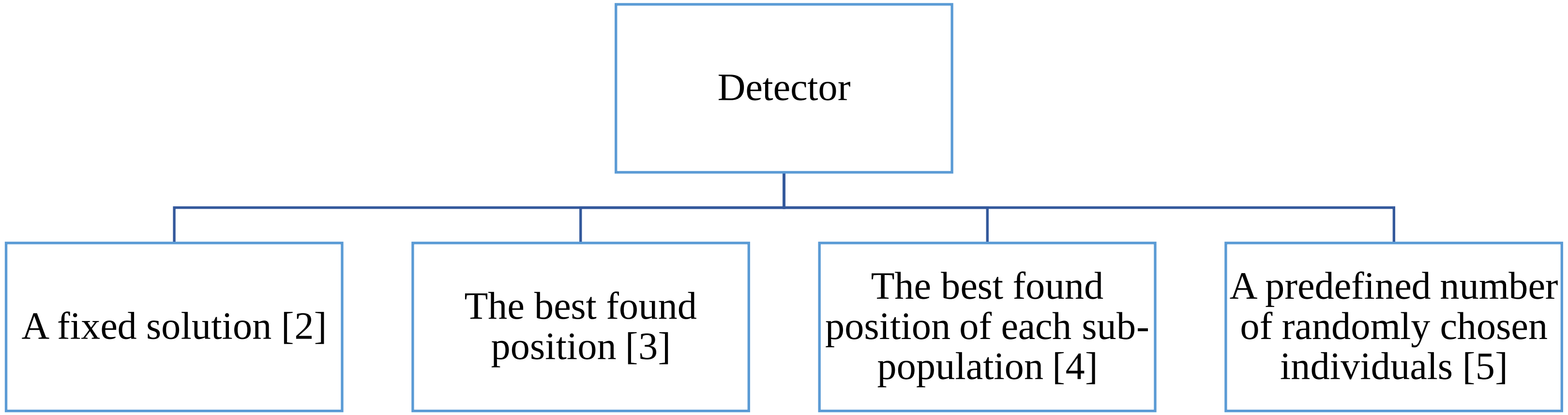
Where $\vec{\boldsymbol{c}}_a$ is the center position of the *a*th sub-population and n_a is its number of individuals.

Detecting convergence by monitoring spatial size of sub-populations is more accurate in comparison to the fitness monitoring based methods. Most EDOs use spatial size monitoring based methods for convergence detection.

[1] X. Li, J. Branke, and T. Blackwell, “Particle swarm with speciation and adaptation in a dynamic environment,” in *Proc. Genet. Evol. Comput. Conf.*, 2006, pp. 51–58.
[2] K. Trojanowski, “Properties of quantum particles in multi-swarms for dynamic optimization,” *Fundamenta Informaticae*, vol. 95, nos. 2–3, pp. 349–380, 2009.
[3] C. Li and S. Yang, “A clustering particle swarm optimizer for dynamic optimization,” in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 439–446.

Change Detection

- Fitness monitoring-based methods
- Reevaluation-based methods
    - A number of solutions, called *detectors*, are reevaluated frequently. If the obtained fitness values are different from the previous values, an environmental change is detected [1].



Detector

A fixed solution [2]

The best found position [3]

The best found position of each sub-population [4]

A predefined number of randomly chosen individuals [5]

[1] J. Branke, “Memory enhanced evolutionary algorithms for changing optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 3, 1999, pp. 1875–1882.

[2] A. Carlisle and G. Dozier, “Adapting particle swarm optimization to dynamic environments,” in *Proc. Int. Conf. Artif. Intell.*, 2000, pp. 429–434.

[3] X. Hu and R. C. Eberhart, “Adaptive particle swarm optimization: Detection and response to dynamic systems,” in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 1666–1670.

[4] T. Blackwell and J. Branke, “Multi-swarm optimization in dynamic environments,” in *Applications of Evolutionary Computing* (Lecture Notes in Computer Science), vol. 3005, G. R. Raidl *et al.*, Eds. Berlin, Germany: Springer, 2004, pp. 489–500.

[5] A. Carlisle and G. Dozler, “Tracking changing extrema with adaptive particle swarm optimizer,” in *Proc. World Autom. Congr.*, 2002, pp. 265–270.

Change Detection: Discussion

- Existing fitness monitoring-based methods may miss an environmental change or detect a false one [1].
- Reevaluation methods are capable of performing *robust 100% detection* if a sufficient number of detectors are used [1].
  - Computationally expensive
  - Cannot be directly used in presence of noise
- In many real-world DOPs, environmental changes are visible and algorithms are informed about them [2].

[1] H. Richter, "Detecting change in dynamic fitness landscapes," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1613–1620.
[2] T. T. Nguyen, "Continuous dynamic optimisation using evolutionary algorithms," Ph.D. dissertation, Sch. Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2011.

Explicit Archiving

- Some obtained information from previous environments is archived and used to accelerate tracking process in the new environment.
  - o This information usually contains the location(s) of promising region(s).

o These components can be reviewed from three different perspective [1]:

1) What solutions to store and when

2) What solutions to delete and when

3)What solutions to retrieve and when

[1] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A survey of evolutionary continuous dynamic optimization over two decades—Part A,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 609 - 629, Aug. 2021.

Explicit Archiving

- What solutions to store and when
  - The best found position by each subpopulation
    - At the end of each environment [1]
    - After convergence [2]

- What solutions to delete and when
  - The oldest archived solution is replaced by the new one [3]
  - The worst archived solution is replaced by the new one [4]
  - The most similar (i.e., the closest one) archived solution is replaced by the new one [5]

Usually, an archive has a finite capacity; therefore, several methods have been designed to remove solutions if the archive is full.

[1] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, “Dynamic optimization using self-adaptive differential evolution,” in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 415–422.
[2] U. Halder, D. Maity, P. Dasgupta, and S. Das, “Self-adaptive cluster-based differential evolution with an external archive for dynamic optimization problems,” in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi, P. N. Suganthan, S. Das, and S. C. Satapathy, Eds. Berlin, Germany: Springer, 2011, pp. 19–26.
[3] Y. G. Woldesenbet and G. G. Yen, “Dynamic evolutionary algorithm with variable relocation,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 500–513, Jun. 2009.
[4] A. M. Turky and S. Abdullah, “A multi-population harmony search algorithm with external archive for dynamic optimization problems,” *Inf. Sci.*, vol. 272, pp. 84–95, Jul. 2014.
[5] T. Zhu, W. Luo, and L. Yue, “Combining multipopulation evolutionary algorithms with memory for dynamic optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 2047–2054.

Explicit Archiving

- What solutions to retrieve and when

Usually, after environmental changes, the archived solutions are used to locate some individuals around the previously discovered promising regions.

  - A predefined number (n) of the best archived solutions are distributed almost uniformly among all m subpopulations and replace their worst individuals [1].
  - A clustering procedure is performed on the archived solutions, and the n best cluster heads (the best solution in a cluster) are distributed almost uniformly among all m subpopulations and replace their worst individuals [1].
  - Each individual is randomized around a solution from the archive that is chosen randomly with a predefined probability [2].
  - In [3], archived solutions participate in the selection process of the parents for crossover in the EA.

Using explicit archiving is suitable *only* for a class of DOPs where the optimum returns to a previous location, or previous environments reappear periodically [4].

[1] T. Zhu, W. Luo, and L. Yue, “Combining multipopulation evolutionary algorithms with memory for dynamic optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 2047–2054.

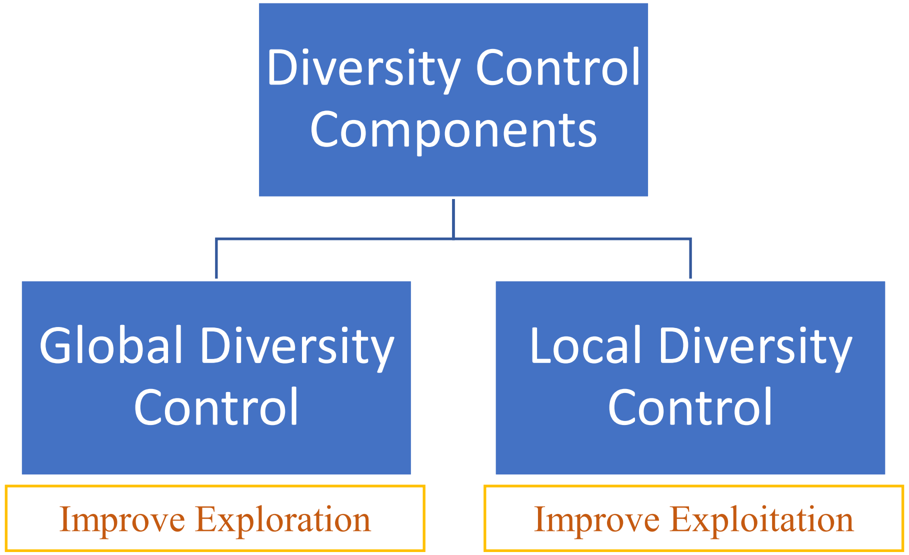
[2] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, “Dynamic optimization using self-adaptive differential evolution,” in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 415–422.

[3] Y. G. Woldesenbet and G. G. Yen, “Dynamic evolutionary algorithm with variable relocation,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 500–513, Jun. 2009.

[4] J. Branke, “Memory enhanced evolutionary algorithms for changing optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 3, 1999, pp. 1875–1882.

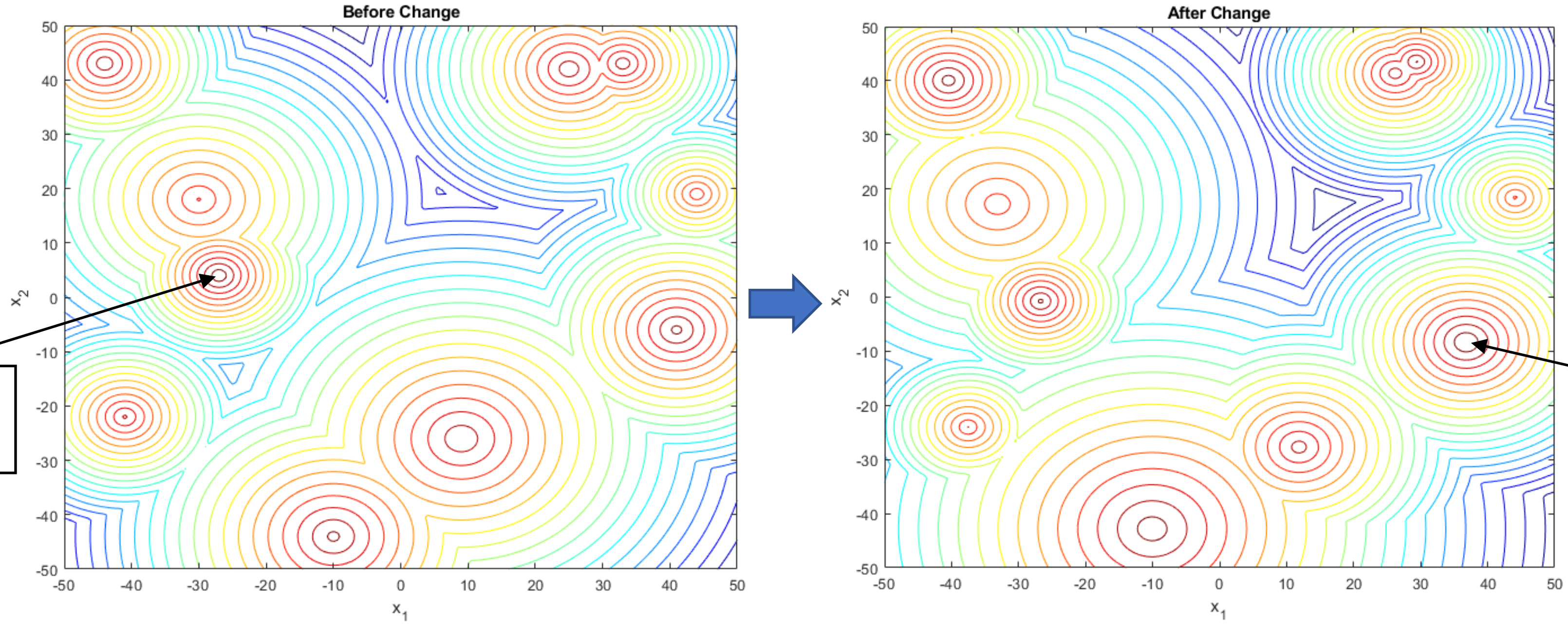
Diversity Control

- Diversity loss is one of the most critical challenges of DOPs and one of the main reasons behind the inefficiency of static optimization algorithms in solving DOPs, because converged populations are incapable of efficiently searching for the new optimum after an environmental change.



Global Diversity Control

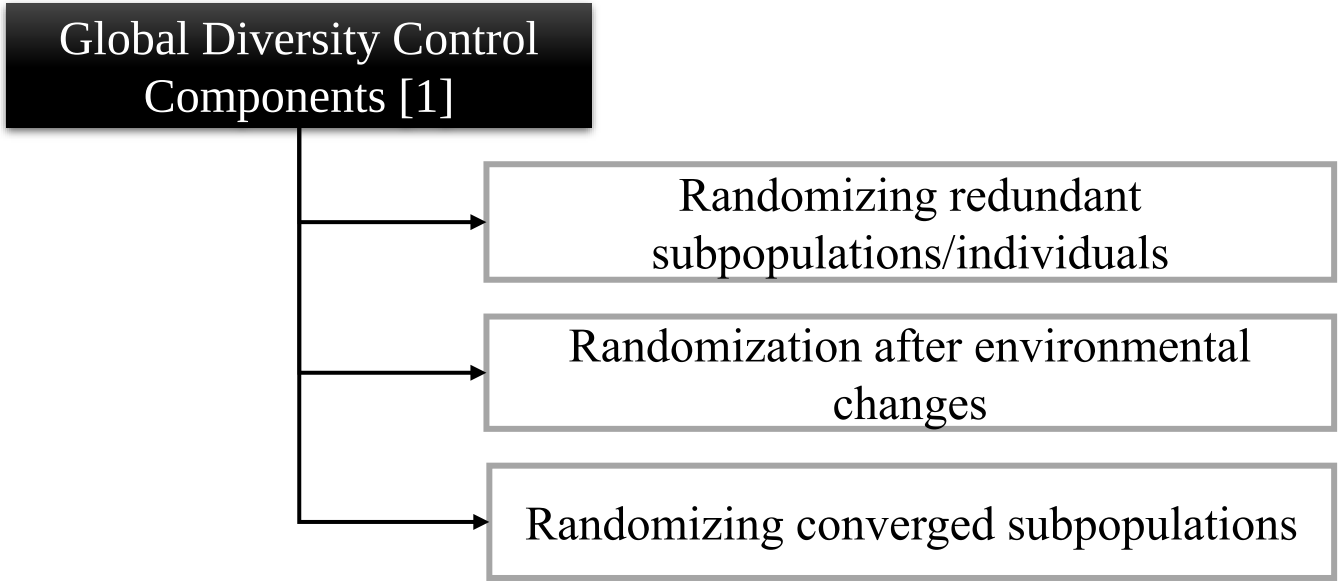
Why do we have to care about global diversity loss in EDOs?



Global optimum position in the *t*th environment

Global optimum position in the (*t*+*1*)th environment

Global Diversity Control



Global Diversity Control Components [1]

Randomizing redundant subpopulations/individuals

Randomization after environmental changes

Randomizing converged subpopulations

[1] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A survey of evolutionary continuous dynamic optimization over two decades—Part A,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 609 - 629, Aug. 2021.

Global Diversity Control

- Randomizing redundant subpopulations/individuals [1]
  - Using one subpopulation with a reasonable number of individuals is usually enough to cover a promising region and tracking it.
  - Exclusion method [2]: if the Euclidean distance between two subpopulations' best found positions becomes less than a predefined radius r_{excl}, then the subpopulation with better best found position is kept and the other one will be randomized.
  - In speciation-based method proposed in [3], subpopulations that reside on the same promising region will be merged since their distance will become closer by converging to the summit. Thereafter, if the number of individuals in a subpopulation is more than a predefined threshold, the redundant inferior individuals are randomized across the search space.

[1] J. Branke, T. Kaussler, C. Schmidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," in *Evolutionary Design and Manufacture*. London, U.K.: Springer, 2000, pp. 299–307.

[2] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing* (Lecture Notes in Computer Science), vol. 3005, G. R. Raidl *et al.*, Eds. Berlin, Germany: Springer, 2004, pp. 489–500.

[3] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1, 2004, pp. 98–103.

Global Diversity Control

- Randomization after environmental changes
  - In the single-population EDO proposed in [1], a predefined number of inferior individuals are randomized after each environmental change.
  - In [2], the best individual of each subpopulation is kept, and the rest are randomized across the search space.
  - In [3], a predefined percentage of the inferior individuals in each subpopulation is randomized across the search space.
  - In [4], subpopulations are divided into two groups of better and worse subpopulations. The individuals of the subpopulation in the worse group are randomized across the search space.

[1] R. C. Eberhart and Y. Shi, “Tracking and optimizing dynamic systems with particle swarms,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 1, 2001, pp. 94–100.

[2] C. Li and S. Yang, “A clustering particle swarm optimizer for dynamic optimization,” in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 439–446.

[3] P. Novoa-Hernández, D. A. Pelta, and C. C. Corona, “Improvement Strategies for Multi-swarm PSO in Dynamic Environments," in *Nature Inspired Cooperative Strategies for Optimization,* J.R. González, D.A. Pelta, C.Cruz, G.Terrazas, and N. Krasnogor, Eds. Berlin, Germany: Springer, 2010, pp. 371–383.

[4] I. Rezazadeh, M. R. Meybodi, and A. Naebi, “Particle swarm optimization algorithm in dynamic environments: Adapting inertia weight and clustering particles,” in *Proc. Eur. Symp. Comput. Model. Simulat.*, 2011, pp. 76–82.

Global Diversity Control

- Randomizing converged subpopulations
  - Used in many EDOs with explicit archiving. When a subpopulation has converged, its best found position is archived, then it will be reinitialized [1].
  - In multi-population EDOs with an adaptive number of subpopulations where a finder subpopulation is used for exploration [2].
  - Anti-convergence component [3]: when all subpopulations have converged, the subpopulation with the worst best found position is randomized.

[1] R. I. Lung and D. Dumitrescu, “A collaborative model for tracking optima in dynamic environments,” in *Proc. IEEE Congr. Evol. Comput.*, 2007, pp. 564–567.

[2] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, “A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization,” *Appl. Soft Comput.*, vol. 13, no. 4, pp. 2144–2158, 2013.

[3] T. Blackwell and J. Branke, “Multiswarms, exclusion, and anticonvergence in dynamic environments,” *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, Aug. 2006.

Global Diversity Control

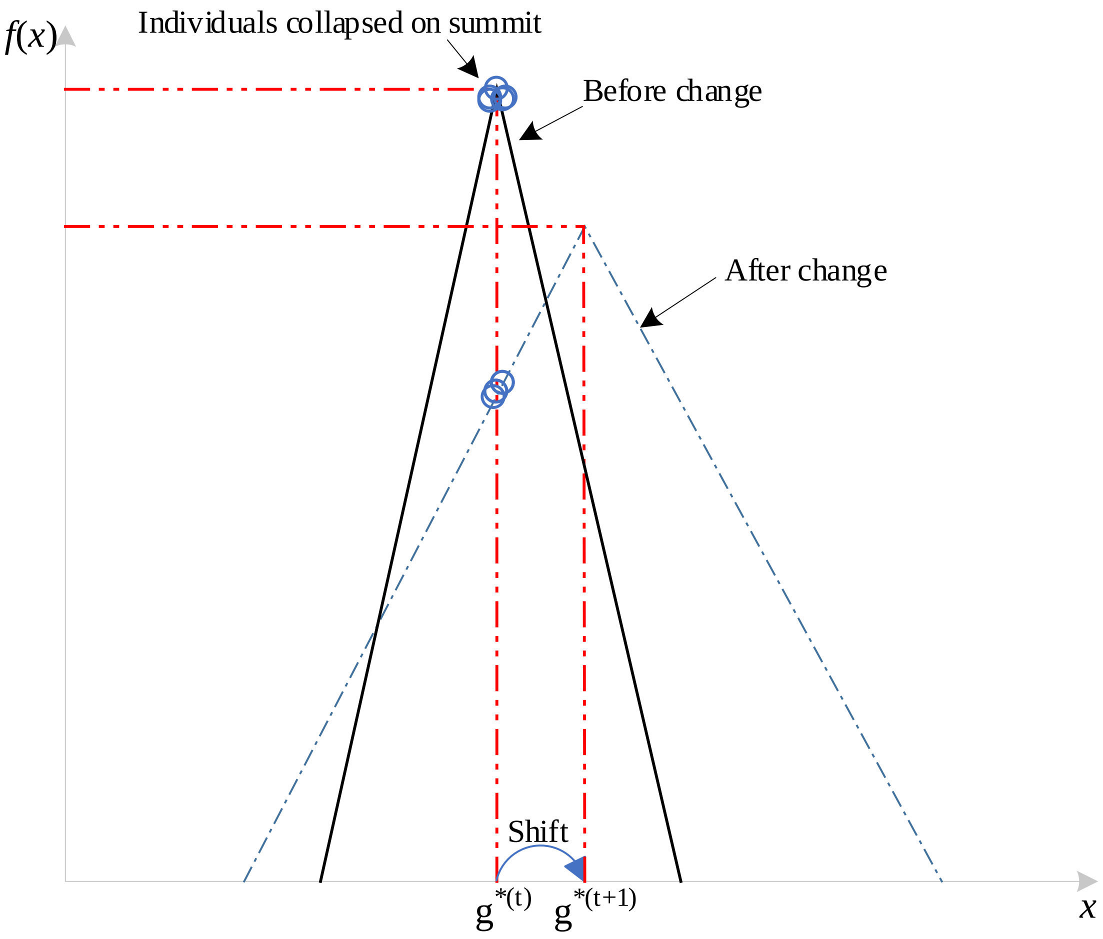
- Randomizing redundant subpopulations/individuals:
  - They are very useful and vital in multi-population EDOs with fixed population size. In such EDOs, using these components also results in saving computational resources.

- Randomization after environmental changes:
  - Using them can result in losing track of the promising regions.

- Randomizing converged subpopulations:
  - They are very useful since they usually increase the global diversity when the algorithm has lost its exploration capability.

Local Diversity Control

Why do we have to care about local diversity loss in EDOs?



Local Diversity Control

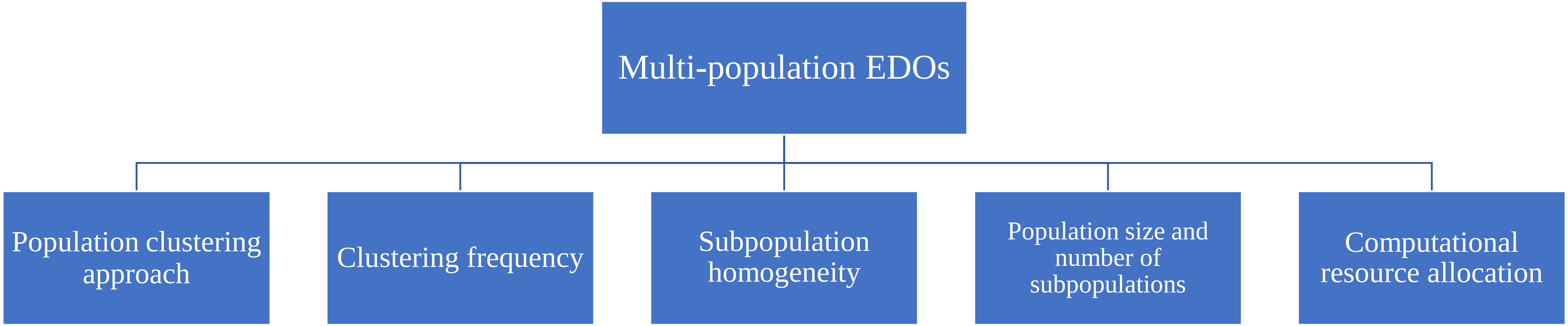
- Maintaining local diversity over time
  - Using quantum particles [1]
  - Adding noise to the position of individuals [2]
  - Another way suggested in the literature is to first allow the local diversity to decrease upto a threshold, then increasing it [3]

- Increasing local diversity after environmental changes
  - Randomization in a Limited Area [4]
- In the first group, a significant amount of the computational resources is used to maintain local diversity over time. The methods in the second group are more effective since they consume considerably less computational resources.

[1] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anticonvergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, Aug. 2006.
[2] R. Mendes and A. S. Mohais, "DynDE: A differential evolution for dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3, 2005, pp. 2808–2815.
[3] S. Bird and X. Li, "Using regression to improve local convergence," in *Proc. IEEE Congr. Evol. Comput.*, 2007, pp. 592–599.
[4] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, "A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 2144–2158, 2013.

Population Division and Management

- Bi-population
  - They use two subpopulations where one is usually used for exploration, and the other is responsible for exploitation.
- Multi-population
  - These EDOs use several subpopulations where the number of subpopulations is a parameter that can be set by either the user or adaptively.
  - They are the most effective and flexible methods to tackle DOPs.



Multi-population EDOs

- Population clustering approach
- Clustering frequency
- Subpopulation homogeneity
- Population size and number of subpopulations
- Computational resource allocation

Population clustering approach

- By index: In this commonly used method, the individuals of each subpopulation are clustered according to their indices [1].
- By Position and Fitness Values: In some multi-population EDOs, the population is divided into subpopulations using clustering methods that work based on the position and fitness value of individuals.
  - *K*-means and fuzzy *c*-means were used for forming sub-populations in [2,3].
    - Number of sub-populations is an input parameter.
    - Fitness of individuals is not considered.
  - In [4], all individuals are listed in a list L. Then, the best individual in L and its $m-1$ nearest individuals are chosen as a subpopulation with the size m, and then all will be removed from L. This process is repeated until L becomes empty.

[1] T. Blackwell and J. Branke, “Multi-swarm optimization in dynamic environments,” in *Applications of Evolutionary Computing* (Lecture Notes in Computer Science), vol. 3005, G. R. Raidl *et al.*, Eds. Berlin, Germany: Springer, 2004, pp. 489–500.
[2] U. Halder, D. Maity, P. Dasgupta, and S. Das, “Self-adaptive cluster-based differential evolution with an external archive for dynamic optimization problems,” in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi, P. N. Suganthan, S. Das, and S. C. Satapathy, Eds. Berlin, Germany: Springer, 2011, pp. 19–26.
[3] R. Mukherjee, G. R. Patra, R. Kundu, and S. Das, “Cluster-based differential evolution with crowding archive for niching in dynamic environments,” *Inf. Sci.*, vol. 267, pp. 58–82, May 2014.
[4] W. Luo, R. Yi, B. Yang, and P. Xu, “Surrogate-assisted evolutionary framework for data-driven dynamic optimization,” *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 2, pp. 137–150, Apr. 2019.

Population clustering approach

- Clustering frequency
  - At the beginning
  - After each environmental change
  - Every iteration
- Homogeneity of subpopulations
  - Homogeneous
  - Heterogeneous
    - Different configurations, such as different sub-population sizes
    - Different optimization component
- Population size and number of subpopulations
  - Fixed population size and number of sub-populations
  - Fixed population size and changing number of sub-populations
  - Changing population size and number of sub-populations

Population clustering approach

- Index-based clustering approaches are easy to implement and without additional computational load. However, they do not consider the attributes of individuals, such as their positions.
- Clustering methods that consider individuals' attributes are computationally heavy as they need to calculate many Euclidean distances. These methods need some input parameters, such as the number of subpopulations, maximum subpopulation size, or subpopulation radius, which affect their flexibility and performance.
- Among multi-population EDOs, those with varying population size and the number of subpopulations are the most efficient. In these EDOs, such numbers are usually adapted to the number of discovered promising regions.

Computational resource allocation

- Computational resource allocation
  - Round Robin/Parallel
  - Deactivating Converged Sub-populations [1]
  - Using a Local Search Operator Around the Best Found Position [2]
  - Subpopulation Size Control [3]
  - Performance-Based Subpopulation Selection [4]
    - Fitness value of the best found position by a sub-population
    - The improvement of subpopulation's best found position in the last iteration that it was active

[1] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A hibernating multiswarm optimization algorithm for dynamic environments," in *Proc. Nat. Biol. Inspired Comput.*, 2010, pp. 363–369.

[2] I. Rezazadeh, M. R. Meybodi, and A. Naebi, "Particle swarm optimization algorithm in dynamic environments: Adapting inertia weight and clustering particles," in *Proc. Eur. Symp. Comput. Model. Simulat.*, 2011, pp. 76–82.

[3] J. Branke, T. Kaussler, C. Schmidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," in *Evolutionary Design and Manufacture*. London, U.K.: Springer, 2000, pp. 299–307.

[4] M. C. du Plessis and A. P. Engelbrecht, "Improved differential evolution for dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 229–234.

- Introduction
- Evolutionary Dynamic Optimization
- **Benchmark Problems**
- Performance Analysis Methods
- Future Research Directions

Benchmark Problems

- To assess the effectiveness of DOAs, they are usually tested on different generated problem instances by DOP benchmarks.
  - Baseline functions
    - Basic static functions
    - The composition of basic static functions [1] (used in GDBG [2])
    - Moving peaks baseline functions
  - Dynamics

[1] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proc. IEEE Swarm Intell. Symp.*, Pasadena, CA, USA, 2005, pp. 68–75.

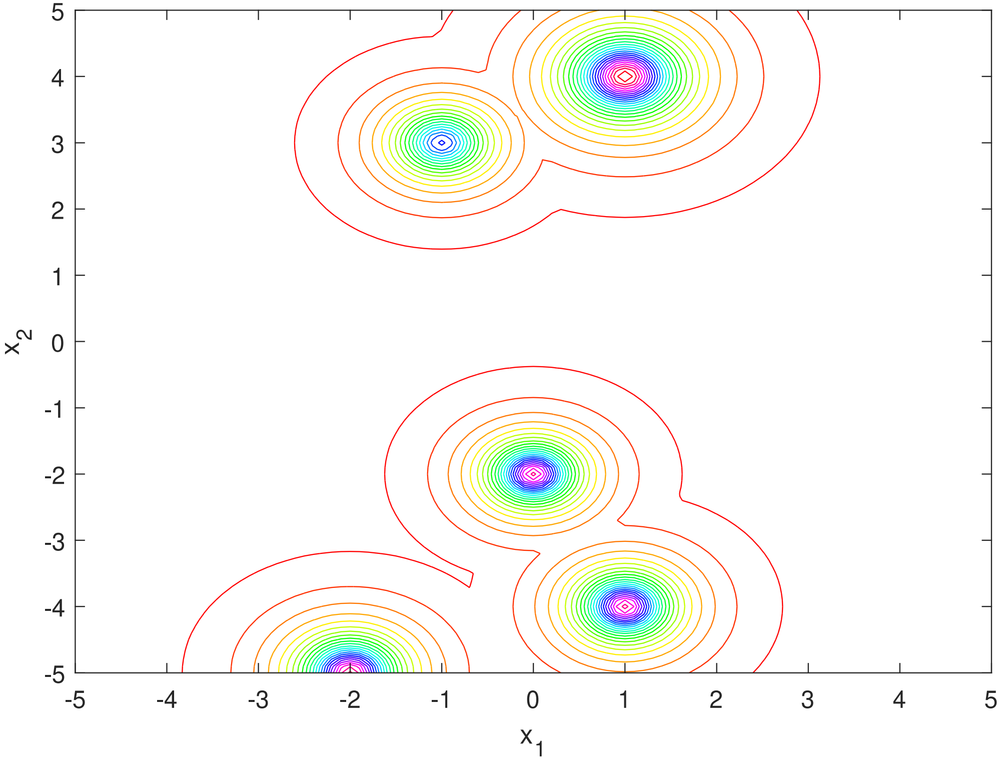
[2] C. Li *et al.*, "Benchmark generator for CEC'2009 competition on dynamic optimization," Center Comput. Intell., Rep., 2008.

Benchmark Problems

Moving peaks benchmark (Scenario 1) [1]

$$f^{(t)}(\vec{x}) = \max_{i\in\{1,\cdots,m\}} \frac{h_i^{(t)}}{1 + w_i^{(t)} \sum_{j=1}^{d} \left(x_j - c_{i,j}^{(t)}\right)^2}$$

Where m is the number of peaks, max(.) defines the basin of attraction of each peak, and $h_i^{(t)}$, $w_i^{(t)}$, and $\vec{c}_i^{(t)}$ are the height, width, and center position of the ith peak in the tth environment, respectively.



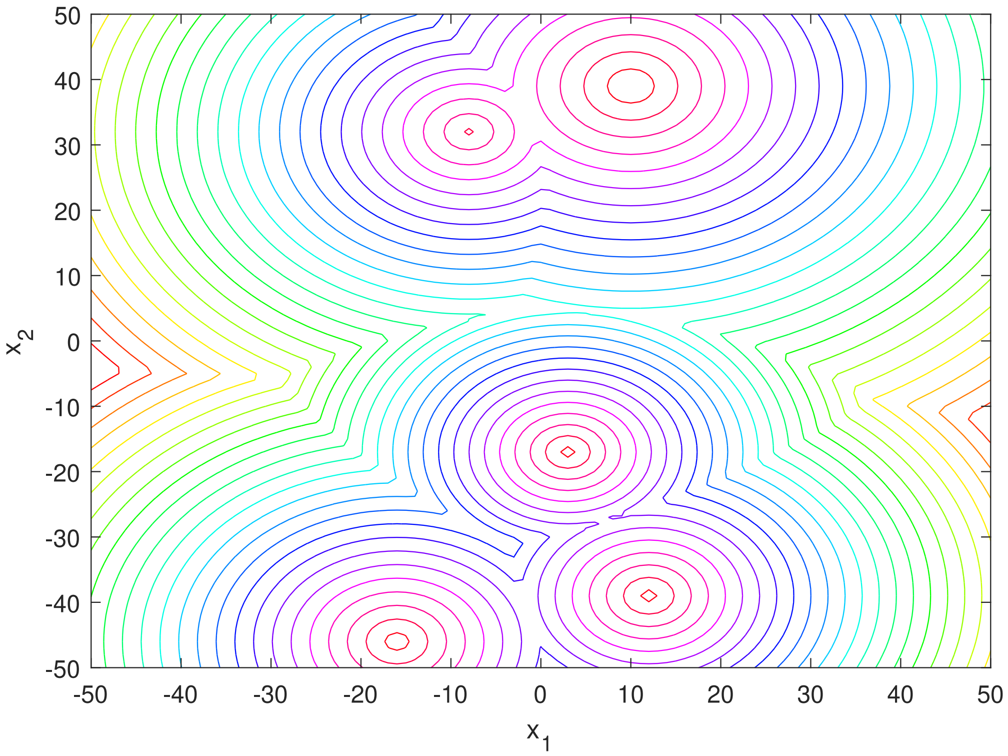
[1] J. Branke, “Memory enhanced evolutionary algorithms for changing optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 3. Washington, DC, USA, 1999, pp. 1875–1882.

Benchmark Problems

DF1 [1] and Moving peaks benchmark (Scenario 2) [1]

$$f^{(t)}(\vec{x}) = \max_{i\in\{1,\cdots,m\}} \left\{ h_i^{(t)} - w_i^{(t)} \left\| \vec{x} - \vec{c}_i^{(t)} \right\| \right\}$$

Where m is the number of peaks, max(.) defines the basin of attraction of each peak, and $h_i^{(t)}$, $w_i^{(t)}$, and $\vec{c}_i^{(t)}$ are the height, width, and center position of the ith peak in the tth environment, respectively.



[1] R. W. Morrison and K. A. D. Jong, “A test problem generator for nonstationary environments,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 3. Washington, DC, USA, 1999, pp. 2047–2053.

[2] J. Branke, “Memory enhanced evolutionary algorithms for changing optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 3. Washington, DC, USA, 1999, pp. 1875–1882.

Benchmark Problems

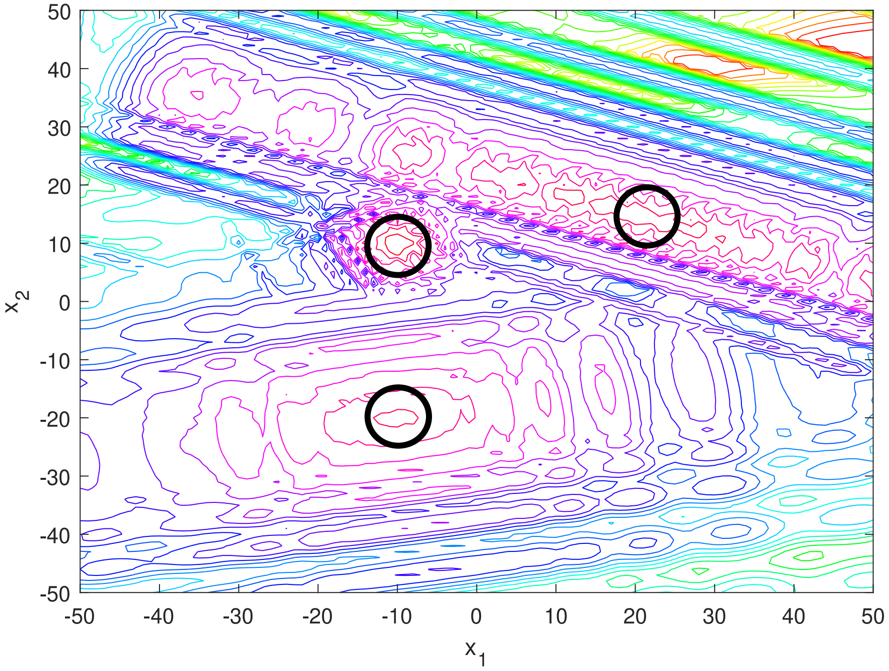
Generalized moving peaks benchmark [1,2]

$$f^{(t)}(\vec{x}) = \max_{i\in\{1,\cdots,m\}}\left\{h_i^{(t)} - \sqrt{\mathbb{T}\left(\left(\vec{x}-\vec{c}_i^{(t)}\right)^{\intercal}\mathbf{R}_i^{(t)\intercal}, i\right)\mathbf{W}_i^{(t)}\mathbb{T}\left(\mathbf{R}_i^{(t)}\left(\vec{x}-\vec{c}_i^{(t)}\right), i\right)}\right\}$$

Where $\mathbb{T}(\vec{y}, i)$: $\mathbb{R}^d \mapsto \mathbb{R}^d$ is calculated as:

$$\mathbb{T}(y_j, i) = \begin{cases} \exp\left(\log(y_j) + \tau_i^{(t)}\left(\sin\left(\eta_{i,1}^{(t)}\log(y_j)\right)\sin\left(\eta_{i,2}^{(t)}\log(y_j)\right)\right)\right) & \text{if } y_j > 0 \\ 0 & \text{if } y_j = 0 \\ -\exp\left(\log(|y_j|) + \tau_i^{(t)}\left(\sin\left(\eta_{i,3}^{(t)}\log(|y_j|)\right)\sin\left(\eta_{i,4}^{(t)}\log(|y_j|)\right)\right)\right) & \text{if } y_j < 0 \end{cases}$$

Where $\mathbf{R}_i^{(t)}$ is the rotation matrix of the *i*th promising region, $\mathbf{W}_i^{(t)}$ is a *d*×*d* diagonal matrix, whose diagonal elements show the width of the *i*th promising region in different dimensions, and $\eta_{i,k\in\{1,2,3,4\}}^{(t)}$ and $\tau_i^{(t)}$ are irregularity parameters of the *i*th promising region.



[1] D. Yazdani, M. N. Omidvar, R. Cheng, J. Branke, T. T. Nguyen, and X. Yao, “Benchmarking continuous dynamic optimization: Survey and generalized test suite,” *IEEE Trans. Cybern.*, early access, Aug. 14, 2020, doi: 10.1109/TCYB.2020.3011828.

[2] D. Yazdani, J. Branke, M. N. Omidvar, C. Li, M. Mavrovouniotis, T. T. Nguyen, S. Yang, and X. Yao, “Generalized moving peaks benchmark,” arXiv preprint arXiv:2106.06174, 2021.

Benchmark Problems

- Landscapes generated by MPB (both scenarios 1 and 2) consist of promising regions that are smooth, symmetric, unimodal, separable, and easy to optimize, which may not be the case in many real-world problems.
- Generated Promising regions by GMPB can range from smooth, unimodal, symmetric, separable, and with circular contour lines to highly irregular, multimodal, asymmetric, fully nonseparable, and ill conditioned.

Benchmark Problems

- Dynamics [1]
  - Small step: $\Delta\phi = \gamma r \tilde{\phi} \|\phi\|$
  - Large step: $\Delta\phi = \tilde{\phi}\|\phi\|\left(\gamma\, \mathrm{sign}(r) + r(\gamma_{\max} - \gamma)\right)$
  - Random: $\Delta\phi = \tilde{\phi}\mathcal{N}(0,1)$
  - Chaotic: $\phi^{(t+1)} = A\phi^{(t)} \frac{1-\phi^{(t)}}{\|\phi\|}$
  - Recurrent: $\phi^{(t+1)} = \phi_{\min} \frac{\|\phi\|}{2}\left(\sin\left(\frac{2\pi}{p}t + \varphi\right) + 1\right)$

Where $\Delta\phi$ is the deviation from the current control parameter values, $\|\phi\|$ is the change range of ϕ, $\phi^{(t)}$ is the offset in the tth environment, $\tilde{\phi} \in (0,1)$ is the change severity of ϕ, $\phi_{\min}$ is the minimum value of ϕ, $\tilde{n} \in (0,1)$ is the noise severity, γ, $\gamma_{\max} \in (0,1)$ and A are constant values, $r \in (0,1)$ is a random number drawn from a uniform distribution, p is the period number, and φ is the initial phase.

  - Recurrent with noise: $\phi^{(t+1)} = \phi_{\min} \frac{\|\phi\|}{2}\left(\sin\left(\frac{2\pi}{p}t + \varphi\right) + 1\right) + \tilde{n}\mathcal{N}(0,1)$

[1] C. Li *et al.*, “Benchmark generator for CEC’2009 competition on dynamic optimization,” Center Comput. Intell., Rep., 2008.

Benchmark Problems

- Test configurations
  - Different numbers of peaks from 1 to 200
  - Different shift severity values from 1 to 5
  - Different dimension numbers from 2 to 10
  - Different change frequencies from 500 to 10000
  - Initial values should be defined randomly
  - Number of environments is usually set to 100

Use the same random seeds for generating problem instances in the experiments.

- Introduction
- Evolutionary Dynamic Optimization
- Benchmark Problems
- Performance Analysis Methods
- Future Research Directions

Performance Analysis Methods

Performance indicators

- Offline error [1]

$$E_o = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta}\left(f^{(t)}\left(\vec{x}^{\circ(t)}\right) - f^{(t)}\left(\vec{x}^{*((t-1)\vartheta+c)}\right)\right)$$

- Best error before change [2]

$$E_{BBC} = \frac{1}{T}\sum_{t=1}^{T}\left(f^{(t)}\left(\vec{x}^{\circ(t)}\right) - f^{(t)}\left(\vec{x}^{*(t)}\right)\right)$$

- Offline performance [3]

$$P_O = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta} f^{(t)}\left(\vec{x}^{*((t-1)\vartheta+c)}\right)$$

where $\vec{x}^{\circ(t)}$ is the global optimum position at the tth environment, T is the number of environments, ϑ is the change frequency, c is the fitness evaluation counter for each environment, $\vec{x}^{*((t-1)\vartheta+c)}$ is the best found position at the cth fitness evaluation in the tth environment, and $\vec{x}^{*(t)}$ is the best found position in the tth environment.

[1] J. Branke and H. Schmeck, “Designing evolutionary algorithms for dynamic optimization problems,” in *Advances in Evolutionary Computing* (Natural Computing Series), A. Ghosh and S. Tsutsui, Eds. Heidelberg, Germany: Springer, 2003, pp. 239–262.

[2] K. Trojanowski and Z. Michalewicz, “Searching for optima in nonstationary environments,” in *Proc. IEEE Congr. Evol. Comput.*, Washington, DC, USA, 1999, pp. 1843–1850.

[3] J. Branke, “Memory enhanced evolutionary algorithms for changing optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, Washington, DC, USA, 1999, pp. 1875–1882.

- Introduction
- Evolutionary Dynamic Optimization
- Benchmark Problems
- Performance Analysis Methods
- **Future Research Directions**

Future Research Directions

Real-world application

- The majority of the literature is focused on artificial benchmark problems and only a few works address real-world DOPs, such as:
  - Training supervised feed-forward neural networks in dynamic classification problems with concept drift [1]
  - Optimizing the hyperparameters of a support vector machine in dynamic environments [2]
  - Optimizing the adaptive farming strategies [3]
  - Solving odor source localization problem [4]
  - Solving contaminant source identification problem [5]
- One extremely important research direction is to formulate objective functions for real-world DOPs and design new DOAs to solve them. An important group of real-world DOPs that have been ignored in the DOP literature are dynamic covering location problems.

[1] A. S. Rakitianskaia and A. P. Engelbrecht, “Training feedforward neural networks with dynamic particle swarm optimisation,” *Swarm Intell.*, vol. 6, no. 3, pp. 233–270, 2012.

[2] D. J. Kalita and S. Singh, “SVM hyper-parameters optimization using quantized multi-PSO in dynamic environment,” *Soft Comput.*, vol. 24, no. 2, pp. 1225–1241, 2020.

[3] N. Jin, M. Termansen, K. Hubacek, J. Holden, and M. Kirkby, “Adaptive farming strategies for dynamic economic environment,” in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 1213–1220.

[4] W. Jatmiko, K. Sekiyama, and T. Fukuda, “A PSO-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment: Theory, simulation and measurement,” *IEEE Comput. Intell. Mag.*, vol. 2, no. 2, pp. 37–51, May 2007.

[5] L. Liu, E. M. Zechman, E. D. Brill, Jr., G. Mahinthakumar, S. Ranjithan, and J. Uber, “Adaptive contamination source identification in water distribution systems using an evolutionary algorithm-based dynamic optimization procedure,” in *Proc. Water Distrib. Syst. Anal. Symp.*, 2008, pp. 1–9.

Future Research Directions

- Automatic Parameter Tuning
  - The majority of components work on the basis of some constant thresholds which need to be tuned for every problem. The performance of such components is highly dependent on the values decided for these thresholds. Besides, the optimal values of these thresholds depend on the characteristics of the current environment, hence, may change over time. This shows the need for components with automatic parameter tuning [1].

- Hyperheuristics
  - An important point in solving DOPs is that if their characteristics change over time, choosing a proper set of components becomes difficult or sometimes even impossible. In such DOPs, hyperheuristic approaches can be used in EDOs to choose the best set of components depending on the current status of the problem [2].

[1] C. Huang, Y. Li, and X. Yao, “A survey of automatic parameter tuning methods for metaheuristics,” *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 201–216, Apr. 2020.

[2] S. A. van der Stockt and A. P. Engelbrecht, “Analysis of selection hyper-heuristics for population-based meta-heuristics in real-valued dynamic optimization,” *Swarm Evol. Comput.*, vol. 43, pp. 127–146, Dec. 2018.

Future Research Directions

- Robust optimization over time (ROOT)
  - The majority of the works in the field are focused on the tracking of a moving optimum, which does not meet the needs of some real-world problems. ROOT can be tackled effectively by extending EDOs that have been originally designed for tracking moving optima [1].
- Constrained DOPs
  - Constrained DOPs are difficult to optimize since their objective function and constraints may change over time. These problems can become more challenging when there are multiple disjointed moving feasible regions [2].
- Large-scale DOPs [3]
  - Large-scale DOPs are difficult problems since the available computational resources in each environment can be very limited to perform optimum tracking in high-dimensional search space. This is even more critical for multi-population EDOs which track multiple moving promising regions using several subpopulations that need a lot of computational resources.

[1] D. Yazdani, T. T. Nguyen, and J. Branke, "Robust optimization over time by learning problem space characteristics," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 143–155, Feb. 2019.

[2] C. Bu, W. Luo, and L. Yue, "Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 14–33, Feb. 2017.

[3] D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, and X. Yao, "Scaling up dynamic optimization problems: A divide-and-conquer approach," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 1–15, Feb. 2020.

Thanks for your attention!

[1] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A survey of evolutionary continuous dynamic optimization over two decades—Part A,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 609 - 629, Aug. 2021.

[2] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A survey of evolutionary continuous dynamic optimization over two decades—Part B,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 630–650, Aug. 2021.