Tracking the Moving Optimum in Dynamic Optimization Problems

Michalis Mavrovouniotis[†] and Danial Yazdani[‡]

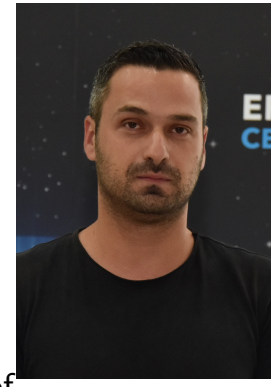
[†]ERATOSTHENES Centre of Excellence, Limassol, Cyprus.
[‡]Data Science Institute, University of Technology Sydney, Sydney, Australia.

June 30, 2024

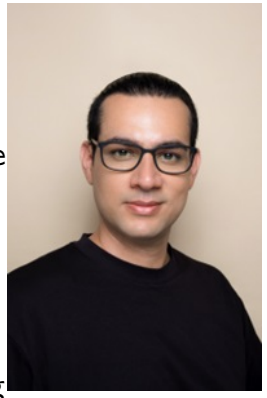
Cyprus
University of
Technology

UTS

Instructors



- **Michalis Mavrovouniotis** received his Ph.D. degree in Computer Science from the University of Leicester, UK (2013). He is currently the Deputy Coordinator of the Big Data Analytics Department at the ERATOSTHENES Centre of Excellence of the Cyprus University of Technology, Limassol, Cyprus. His research interests include evolutionary computation, neural computation, swarm intelligence, memetic computing, combinatorial optimization problems, heuristic and metaheuristics, artificial intelligence in dynamic and uncertain environments, and relevant real-world applications. He has contributed to several funded projects (with partners in both academia and industry). Dr. Mavrovouniotis has more than 70 peer-reviewed publications (with h-index of 26 according to Google Scholar) and he has been included in the Stanford University World's Top 2% Scientists list published in October 2023. He is currently the Chair of the IEEE Task Force on Evolutionary Computation in Dynamic and Uncertain Environments, a member of the IEEE Computational Intelligence Society and an Associate Fellow of the UK Higher Education Academy.



- **Danial Yazdani** received his Ph.D. degree in computer science from Liverpool John Moores University, Liverpool, U.K., in 2018. He is currently a Research Fellow at the Data Science Institute, University of Technology Sydney. Prior to that, he was a Research Assistant Professor with the Department of Computer Science and Engineering at the Southern University of Science and Technology, Shenzhen, China. His primary research interests include learning and optimization in dynamic environments, where he has contributed as the first author in over 20 peer-reviewed publications in this field, nine of which were published in prestigious IEEE/ACM Transactions. Dr. Yazdani was a recipient of the 2023 IEEE Computational Intelligence Society Outstanding PhD Dissertation Award, the Best Thesis Award from the Faculty of Engineering and Technology at Liverpool John Moores University, and the SUSTech Presidential Outstanding Postdoctoral Award from Southern University of Science and Technology.

Overview

What are Dynamic Optimization Problems (DOPs)

- In general, DOPs are problems that change over time (Jin & Branke 2005*b*)
- A DOP can be defined as follows:

$$F = f(\mathbf{x}, \phi, t)$$

  - F: is the optimization problem
  - f: is the cost function
  - $\mathbf{x}$: feasible solution
  - ϕ: control parameter(s)
  - t: time index
- Changes may involve:
  - Objective function
  - Problem instance
  - Constraints
  - ...

What are the characteristics?

- Key characteristics:
  - Frequency
  - Magnitude
- Classification criteria:
  - Time-linkage: Does the future behaviour of the problem depend on the current solution?
  - Predictability: Are changes predictable?
  - Visibility: Are changes visible or detectable?
  - Cyclicity: Are change recurrent in the search space?

Why we should study DOPs?

- Many real-world problems are actually DOPs, e.g.,
- Vehicle routing problem: find the optimal routes for a fleet of vehicles to deliver goods
  - Traffic jams
  - New order may arrive
  - Vehicle may break down
  - Customer demands may change
  - Other uncertainties...

Dynamic Benchmark Generators

Dynamic Test Environments

- Benchmark generators are essential tools, due to the limited theoretical work (Mavrovouniotis et al. 2012)
  - Develop new algorithms
  - Empirically analyze/investigate the behaviour of algorithms in DOPs
  - Testbed to compare the performance with existing algorithms
- Key idea: model different real-world scenarios
- Sometimes it is sacrificed for the sake of benchmarking

Dynamic Test Environments – cont'd

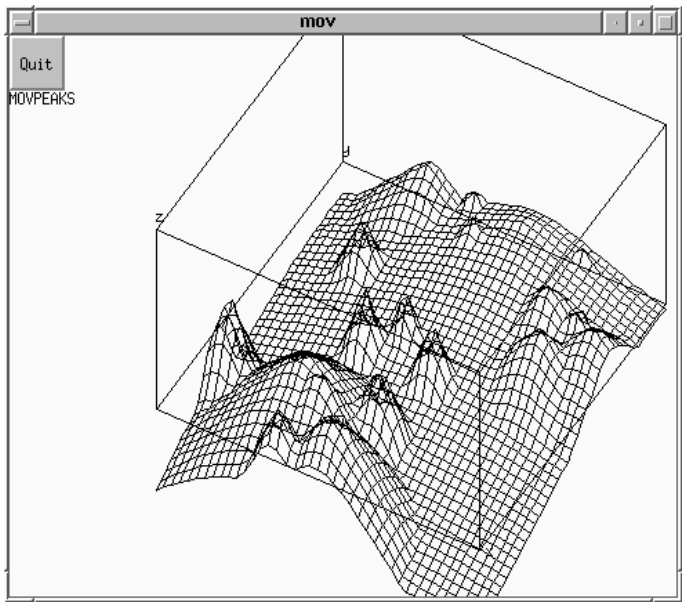
- Basic principle ⇒ change the base of static problem(s)
- Real space:
  - Switch between different functions
  - Move/reshape peaks in the fitness landscape (Branke 1999)
- Binary space:
  - Switch between 2 or more states of a problem
  - Use binary masks (Yang & Yao 2005)
- Combinatorial space:
  - Change decision variables
  - Add or delete decision variables (Guntsch & Middendorf 2002)

Moving Peaks Benchmark (MPB) Problem

- The MPB problem (Branke 1999) in the d-dimensional space:

$$f(\mathbf{x},t) = \max_{i=1,\ldots,p} \frac{H_i(t)}{1 + W_i(t)\sum_{j=1}^{d}(x_j(t) - X_{ij}(t))^2}$$

  - $W_i(t), H_i(t), X_i(t) = \{X_{i1}, \ldots, X_{id}\}$: height, width, location of peak i at t.



- The dynamics:

$$H_i(t) = H_i(t-1) + height\ severity * \sigma$$

$$W_i(t) = W_i(t-1) + width\ severity * \sigma$$

$$\mathbf{X_i}(t) = \mathbf{X_i}(t-1) + \mathbf{v_i}$$

$$\mathbf{v_i}(t) = \frac{s}{|\mathbf{r} + \mathbf{v_i}(t-1)|}((1-\lambda)\mathbf{r} + \lambda\mathbf{v_i}(t-1))$$

  - $\sigma : \mathcal{N}(0,1); \lambda$: correlated parameter; $\mathbf{v_i}(t)$: shift vector, combines random vector $\mathbf{r}$ and $\mathbf{v_i}(t-1)$ is normalized to the shift length s

Dynamic Knapsack Problems (DKPs)

- Static Knapsack Problem:
  - Given n items, each with a weight and a profit, and a knapsack with a fixed capacity, select items to fill up the knapsack to maximize the profit while satisfying the knapsack capacity constraint.
- The DKP:
  - Generated by changing weights and profits of items, and/or knapsack capacity over time as:

$$\max f(\mathbf{x}, t) = \sum_{i=1}^{n} p_i(t) \cdot x_i(t), \quad s.t. : \sum_{i=1}^{n} w_i(t) \cdot x_i(t) \leq C(t)$$

  - $\mathbf{x}(t) \in \{0,1\}^n$: a solution at time t
  - $x_i(t) \in \{0,1\}$: indicates whether item i is included or not
  - $p_i(t)$ and $w_i(t)$: profit and weight of item i at t
  - $C(t)$: knapsack capacity at t

Exclusive OR (XOR) DOP Generator

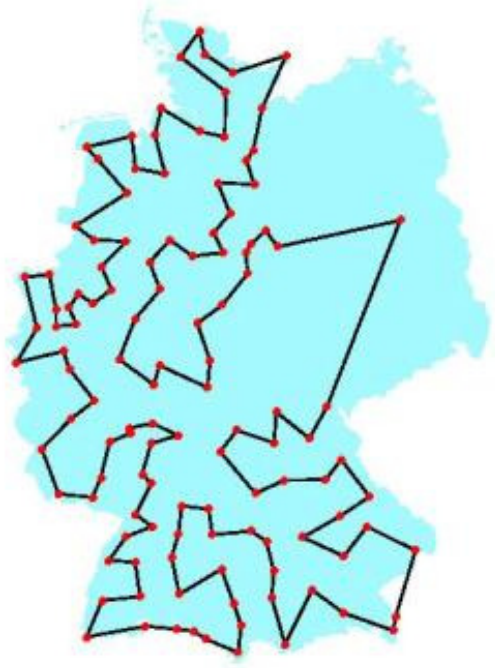
- Generates a DOP from any static binary $f(\mathbf{x})$ (Yang & Yao 2005)
- '$\oplus$' is the XOR operator (i.e., $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 0 = 0$)
- Suppose the environment changes every τ generations
- For each environmental period $k = \lceil t/\tau \rceil$, do:
  1. Create a binary template $\mathbf{T}(k)$ with mn ones randomly
  2. Create a XORing mask $\mathbf{M}(k)$ incrementally
     $\mathbf{M}(k) = \mathbf{M}(k-1) \oplus \mathbf{T}(k)$
  3. Evaluate as follows: $f(\mathbf{x}, t) = f(\mathbf{x} \oplus \mathbf{M}(k))$
- τ and m control the frequency and magnitude of change, respectively

Dynamic Travelling Salesperson Problems

- Static TSP:
  - Given a set of cities, find the shortest route that visits each city once and only once.
- Dynamic TSP (DTSP):
  - The aim is to find the minimum-cost Hamiltonian route at time t

$$\min f(\mathbf{x}, t) = d_{x_n, x_1}(t) + \sum_{i=1}^{n-1} d_{x_i, x_{i+1}}(t)$$

  - n is the number of cities
  - $d_{ij}(t)$ cost from city i to j



DTSP: Node Changes

- Modelling the arrival of new orders (Guntsch & Middendorf 2002)
- Let $N_{in}(k)$ be the current working node set
- Let $N_{out}(k)$ be a randomly generated node set
- Every τ iterations, exactly $\lceil mn \rceil$ nodes are randomly selected from $N_{out}(k)$ to replace exactly $\lceil mn \rceil$ number of nodes randomly selected in $N_{in}(k)$

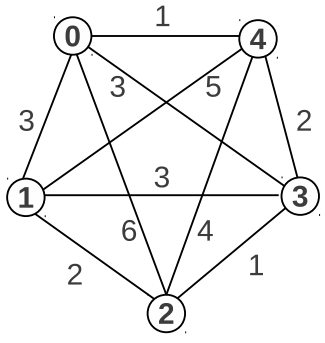
DTSP: Weight Changes

- Modeling traffic changes (Tinós et al. 2014)
- A problem instance consists of several arcs with particular weights
- Increase/decrease arcs' weights
  - Randomly generated from normal distribution
- Exactly $\lceil mn(n-1) \rceil$ arcs are selected to modify their weight

Dynamic Permutation Benchmark Generator (DPBG)

- Key idea: Swap nodes that cause a change to the encoding of the problem instance (Mavrovouniotis et al. 2012)
- Two vectors of $\lceil mn \rceil$ cities are randomly generated
- Exactly $\lceil mn \rceil$ pairwise swaps are performed
- The optimal solution will have different encoding but the same value

Optimum -> (0,4,3,2,1,0) = **9**

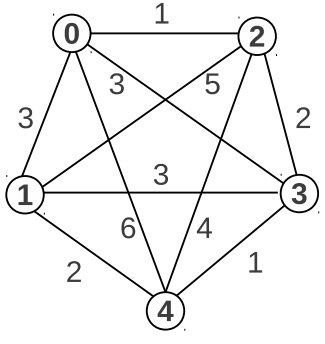


Distance matrix before change

	0	**1**	**2**	**3**	**4**
0	0	3	6	5	1
1	3	0	2	3	3
2	6	2	0	1	4
3	5	3	1	0	2
4	1	3	4	2	0

Swap City Location (4,2)

Optimum -> (0,2,3,4,1,0) = **9**



Distance matrix after change

	0	**1**	**2**	**3**	**4**
0	0	3	1	5	6
1	3	0	3	3	2
2	1	3	0	2	4
3	5	3	2	0	1
4	6	2	4	1	0

Performance Measures

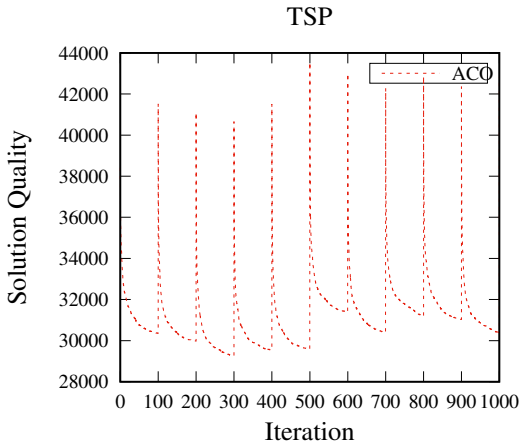
- For static problem there are 2 key performance measures
  - Deviation from the optimal
  - Convergence speed
- For DOPs, there are several measures (Nguyen et al. 2012)
  - Optimality-based performance measures, e.g.,
    - Offline performance and error
    - Best before change
    - ...
  - Behaviour-based performance measures, e.g.,
    - Diversity
    - Robustness
    - ...

Performance Measures – (cont'd)

- Offline performance and error (Branke 1999, Branke & Schmeck 2003):

$$P_{offline} = \frac{1}{E}\sum_{t=1}^{E} f(\mathbf{x}^{\mathbf{best}}, t)$$

$$P_{error} = \frac{1}{E}\sum_{t=1}^{E} \left(f(\mathbf{x}^{*}, t) - f(\mathbf{x}^{\mathbf{best}}, t)\right)$$



  - E: total number of observations
  - $\mathbf{x}^{\mathbf{best}}$ and $\mathbf{x}^{*}$: best so far and optimal solution
- Best before change (Trojanowski & Michalewicz 1999):

$$P_{change} = \frac{1}{M}\sum_{k=1}^{M} f(\mathbf{x}^{\mathbf{best}}, k\tau - 1)$$

  - M: total number of changes
  - τ: frequency of change

Performance Measures – (cont'd)

- Robustness (Yu et al. 2009):

$$P_{robust} = \frac{1}{M-1} \sum_{k=1}^{M-1} \begin{cases} 1, & \text{if } \frac{f(\mathbf{x}^{\mathbf{best}}, k\tau - 1)}{f(\mathbf{x}^{\mathbf{best}}, k\tau)} > 1, \\ \frac{f(\mathbf{x}^{\mathbf{best}}, k\tau - 1)}{f(\mathbf{x}^{\mathbf{best}}, k\tau)}, & \text{otherwise.} \end{cases}$$

- Diversity:

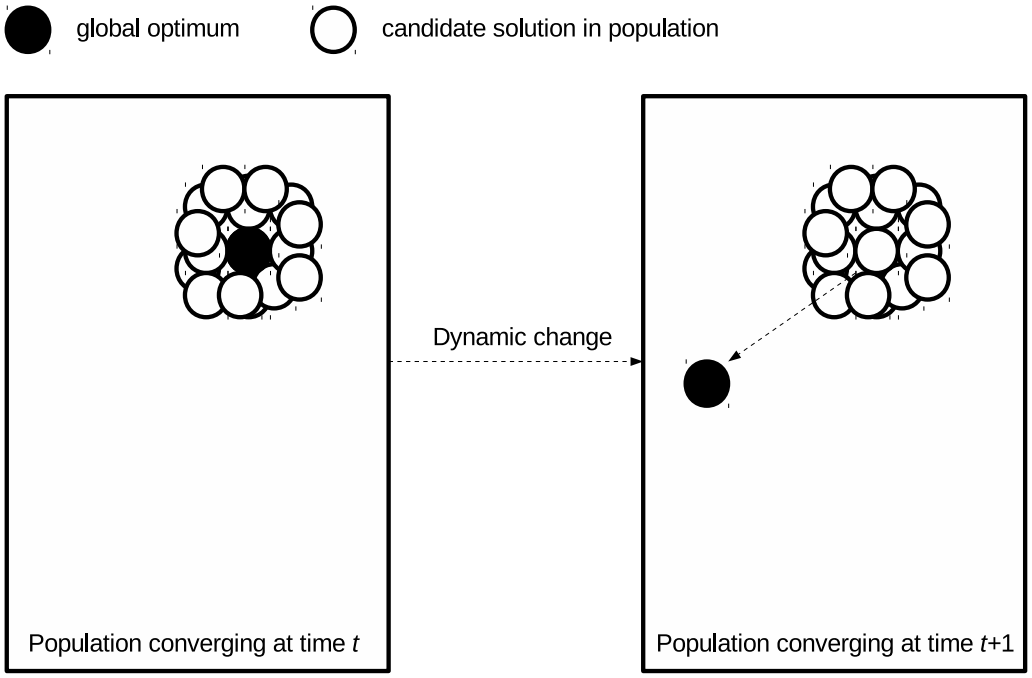
$$P_{diversity} = \frac{1}{\omega(\omega - 1)} \sum_{p=1}^{\omega} \sum_{q \neq p}^{\omega} DIV(p, q)$$

  - ω: number of individuals
  - $DIV(p, q)$: diversity metric between individual p and q, e.g.,
    - DTSP: common edges
    - DKP: Hamming distance
    - MPB: Euclidean distance

Dynamic Optimization Algorithms

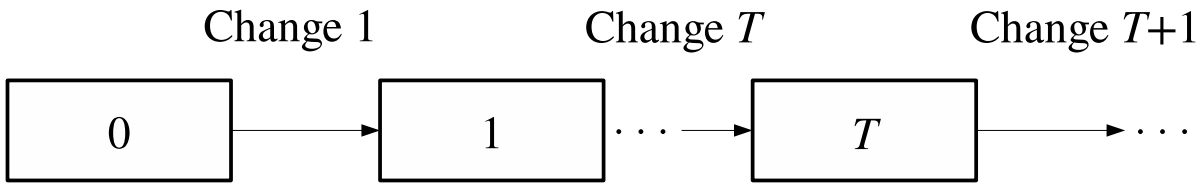
Why DOPs are challenging?

- Hard to escape from previously converged optimum
- Loss of diversity
- Outdated memory
- Limited computational resources



How to address dynamic changes?

- A straight forward method is to consider every change as the arrival of a new problem that needs to be solved from scratch



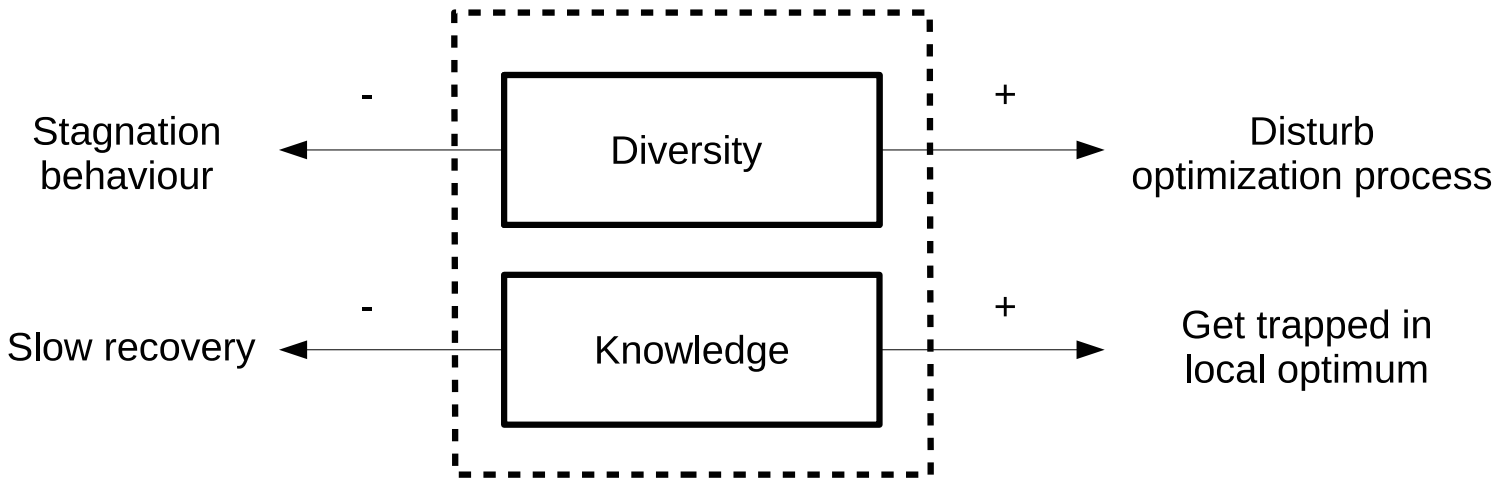
- A more efficient way is to somehow reuse information (Jin & Branke 2005*a*)
- Evolutionary computation and swarm intelligence are good tools because of their intrinsic characteristics (Mavrovouniotis et al. 2020, Yang et al. 2013)
  - Genetic algorithms
  - Ant colony optimization
  - Particle swarm optimization
  - Differential evolution
  - Estimation of distribution algorithms
  - ...

Why optimizers have to be enhanced?

- DOPs challenge all optimizers!
- Have been designed to locate static optimum quickly and precisely
- However, DOPs require continuous tracking of the moving optimum
- Solution $\Rightarrow$ design strategies to enhance their adaptation capabilities

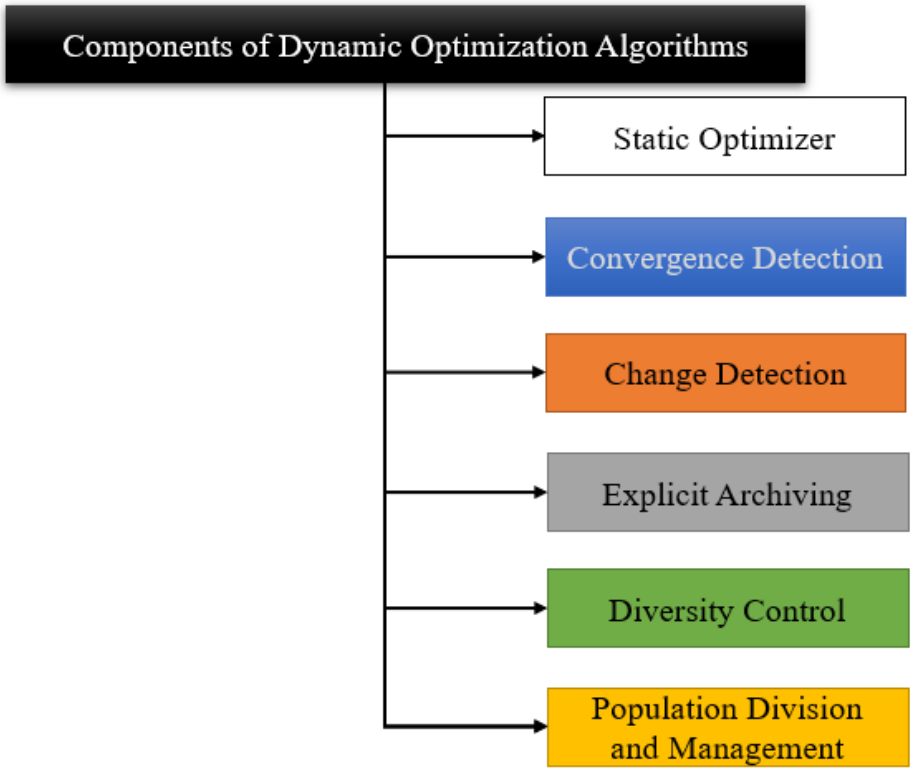
Enhancing Optimizers for DOPs

- They have to balance two unrelated but conflicting factors (Mavrovouniotis et al. 2017)



Dynamic Optimization Algorithms: Components

- Consist of several components (Yazdani et al. 2021)
- Crucial component is the optimizer
- Some components run every iteration
- Other components are triggered when some certain conditions are met



Convergence Detection – Fitness-based

- Can be detected by monitoring the fitness of the best so far solution (Plessis & Engelbrecht 2012)

$$\begin{cases} \text{converged} & f(\mathbf{x}^{\mathbf{best}}, t) = f(\mathbf{x}^{\mathbf{best}}, t-i) \\ \text{uncoverged} & f(\mathbf{x}^{\mathbf{best}}, t) > f(\mathbf{x}^{\mathbf{best}}, t-i) \end{cases}$$

  - $\mathbf{x}^{\mathbf{best}}$ is the best solution found at iteration t
  - Can detect convergence only if it has not improved during the last i iterations

- Improved flexibility (Yazdani et al. 2013):

$$\begin{cases} \text{converged} & f(\mathbf{x}^{\mathbf{best}}, t) - f(\mathbf{x}^{\mathbf{best}}, t-i) \leq \epsilon \\ \text{uncoverged} & f(\mathbf{x}^{\mathbf{best}}, t) - f(\mathbf{x}^{\mathbf{best}}, t-i) > \epsilon \end{cases}$$

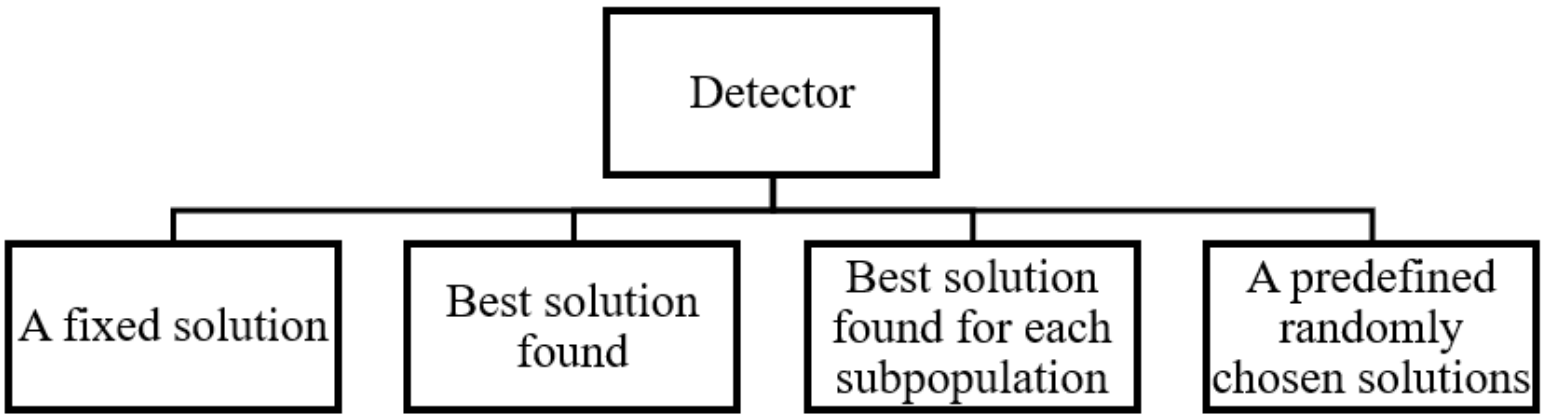
- where ϵ is a small constant

Convergence Detection – Spatial size-based

- Can be also monitored by measuring the spatial size of a population
  - Euclidean distance ⇒ real space
  - Hamming distance ⇒ binary space
- Spatial size can be defined as:
  - Maximum distance from any candidate solution to the best solution found (Li et al. 2006)
  - The largest distance between any two candidate solutions (Trojanowski 2009)
- Convergence is detected if the spatial size falls below a predefined threshold
- **Spatial size-based** convergence generally is more accurate than **fitness-based** convergence monitoring

Change Detection

- Most popular method is the use of *detectors* (Branke 1999)
- Candidate solution that is reevaluated frequently
- If the obtained fitness values are different from the previous values; then a change is detected.

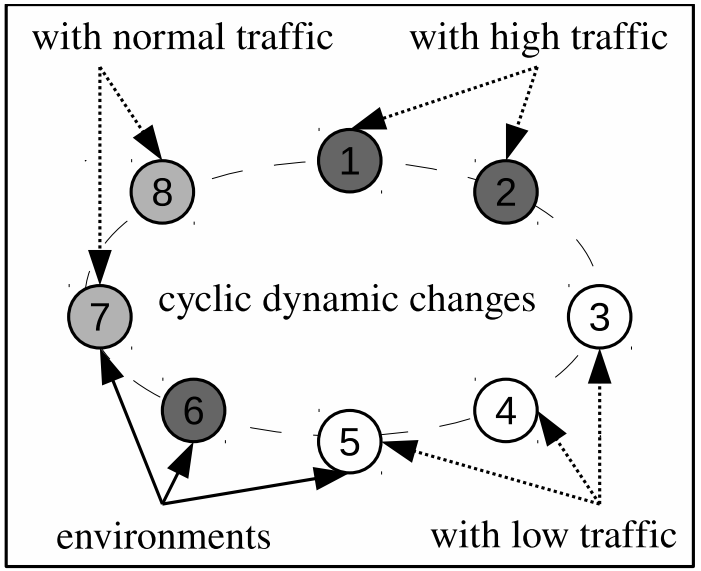


Change Detection – cont'd

- Detectors are capable of performing robust detection (Richter 2009)
  - Computational expensive, if overuse
  - Fail in the presence of noise
- In many real-world applications, environmental changes are visible
- Optimizers are usually informed about the changes

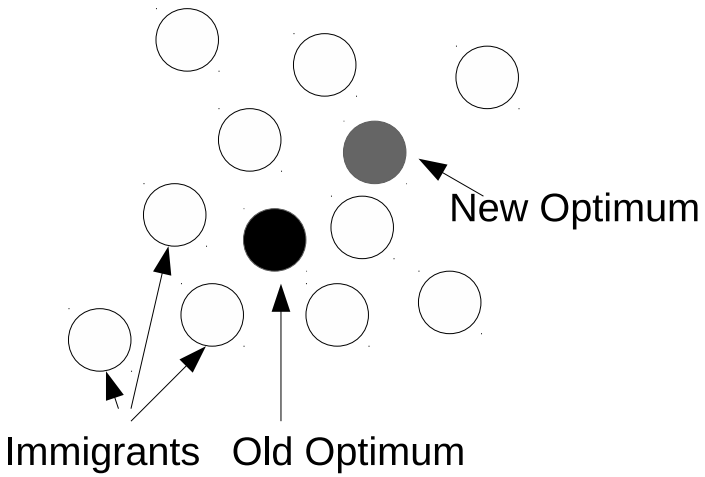
Explicit Archiving

- Previous knowledge is stored in memory structure (Yang 2008)
- Useful in cyclic environments
- There are three main concerns:
  1. What solutions to store and when?
  2. What solutions to delete and when?
  3. What solutions to retrieve and when?



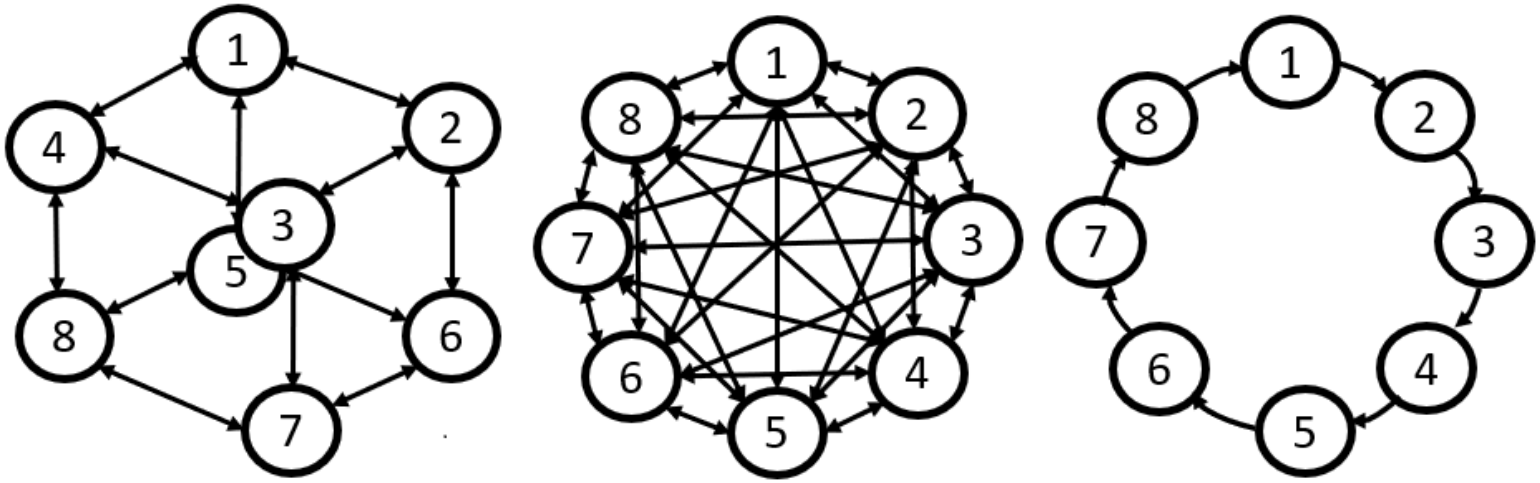
Diversity Control

- Diversity loss is the main reason behind the inefficiency of optimizers in solving DOPs.
  - Maintain diversity during the searching process (Yang 2007)
  - Increase diversity after the change (Yazdani et al. 2013)
  - Reinitialize converged subpopulations (Blackwell & Branke 2006)



Population Division and Management

- Multi-population schemes use co-operating subpopulations
- They are the most effective and flexible methods to address DOPs
- Several design characteristics to decide:
  - Subpopulation homogeneity (Branke et al. 2000)
  - Number of subpopulations and their size (Li et al. 2016)
  - Information exchange strategy (Whitley et al. 1998)
  - Computational resource allocation



Future Work

EDOLAB: An open source MATLAB Plarform

- A platform focused on DOPs with continuous search space
- EDOLAB contains:
  - More than 25 dynamic optimization algorithms
  - Two performance indicators
    - Offline performance
    - Best before change
  - Four dynamic benchmark generators:
    - MPB (Branke 1999)
    - Generalized Moving Peaks Benchmark (GMPB) (Yazdani et al. 2022)
    - Generalized Dynamic Benchmark Generator (GDBG) (Li et al. 2009)
    - Free Peaks (FPs) (Li et al. 2019)
- For more information:
  `https://github.com/Danial-Yazdani/EDOLAB-MATLAB`
- Plan to extend it for discrete search space

Future Work

- Adaptive parameter tuning over time
  - Different tuning for different environments
- Dynamic multi(and many)-objective problems: deserve more effort
- Incorporate machine learning techniques
  - Useful in estimating computational expensive objectives
  - Predict changes to prepare the algorithm for efficient adaptation

Relevant Information

- IEEE CIS Task Force on EC in Dynamic and Uncertain Environments (ECiDUE)
  - `https://mavrovouniotis.github.io/ieee-tf-ecidue/`
  - Maintained by Michalis Mavrovouniotis
- IEEE Competitions
  - `https://competition-hub.github.io/GMPB-Competition/`
- IEEE Special Sessions:
  - Every year since 2004
  - `https://mavrovouniotis.github.io/ECiDUE24/`

References I

Blackwell, T. & Branke, J. (2006), 'Multiswarms, exclusion, and anti-convergence in dynamic environments', *IEEE Transactions on Evolutionary Computation* **10**(4), 459–472.

Branke, J. (1999), Memory enhanced evolutionary algorithms for changing optimization problems, *in* 'Proceedings of the 1999 Congress on Evolutionary Computation', Vol. 3, pp. 1875–1882.

Branke, J., Kaussler, T., Smidt, C. & Schmeck, H. (2000), A multi-population approach to dynamic optimization problems, *in* I. Parmee, ed., 'Evolutionary Design and Manufacture', Springer London, pp. 299–307.

Branke, J. & Schmeck, H. (2003), Designing evolutionary algorithms for dynamic optimization problems, *in* A. Ghosh & S. Tsutsui, eds, 'Advances in Evolutionary Computing', Natural Computing Series, Springer Berlin Heidelberg, pp. 239–262.

Guntsch, M. & Middendorf, M. (2002), Applying population based ACO to dynamic optimization problems, *in* M. Dorigo, G. Di Caro & M. Sampels, eds, 'Ant Algorithms', Vol. 2463 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 111–122.

Jin, Y. & Branke, J. (2005*a*), 'Evolutionary optimization in uncertain environments–a survey', *IEEE Transactions on Evolutionary Computation* **9**(3), 303–317.

Jin, Y. & Branke, J. (2005*b*), 'Evolutionary optimization in uncertain environments: a survey', *IEEE Transactions on Evolutionary Computation* **9**(3), 303–317.

Li, C., Nguyen, T. T., Yang, M., Mavrovouniotis, M. & Yang, S. (2016), 'An adaptive multi-population framework for locating and tracking multiple optima', *IEEE Transactions on Evolutionary Computation* **20**(4), 590–605.

Li, C., Nguyen, T. T., Zeng, S., Yang, M. & Wu, M. (2019), 'An open framework for constructing continuous optimization problems', *IEEE Transactions on Cybernetics* **49**(6), 2316–2330.

Li, C., Yang, S., Nguyen, T., Yu, E., Yao, X., Jin, Y., Beyer, H.-G. & Suganthan, P. (2009), Benchmark generator for cec'2009 competition on dynamic optimization, Technical report, Department of Computer Science, University of Leicester, U.K.

Li, X., Branke, J. & Blackwell, T. (2006), Particle swarm with speciation and adaptation in a dynamic environment, *in* 'Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation', GECCO '06, ACM, New York, NY, USA, pp. 51–58.

References II

Mavrovouniotis, M., Li, C. & Yang, S. (2017), 'A survey of swarm intelligence for dynamic optimization: Algorithms and applications', *Swarm and Evolutionary Computation* **33**, 1–17.

Mavrovouniotis, M., Yang, S., Van, M., Li, C. & Polycarpou, M. (2020), 'Ant colony optimization algorithms for dynamic optimization: A case study of the dynamic travelling salesperson problem [research frontier]', *IEEE Computational Intelligence Magazine* **15**(1), 52–63.

Mavrovouniotis, M., Yang, S. & Yao, X. (2012), A benchmark generator for dynamic permutation-encoded problems, *in* C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia & M. Pavone, eds, 'Parallel Problem Solving from Nature - PPSN XII', Vol. 7492 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 508–517.

Nguyen, T., Yang, S. & Branke, J. (2012), 'Evolutionary dynamic optimization: A survey of the state of the art', *Swarm and Evolutionary Computation* **6**, 1–24.

Plessis, M. & Engelbrecht, A. (2012), 'Differential evolution for dynamic environments with unknown numbers of optima', *Journal of Global Optimization* pp. 1–27.

Richter, H. (2009), Detecting change in dynamic fitness landscapes, *in* '2009 IEEE Congr. Evol. Comput.', pp. 1613–1620.

Tinós, R., Whitley, D. & Howe, A. (2014), Use of explicit memory in the dynamic traveling salesman problem, *in* 'Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation', GECCO'14, ACM, New York, NY, USA, pp. 999–1006.

Trojanowski, K. (2009), 'Properties of quantum particles in multi-swarms for dynamic optimization', *Fundamenta Informaticae* **95**(2-3), 349–380.

Trojanowski, K. & Michalewicz, Z. (1999), Searching for optima in non-stationary environments, *in* 'Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on', Vol. 3, pp. 1843–1850.

Whitley, D., Rana, S. & Heckendorn, R. B. (1998), 'The Island Model Genetic Algorithm: On Separability, Population Size and Convergence', *Journal of Computing and Information Technology* **7**, 33–47.

Yang, S. (2007), Genetic algorithms with elitism-based immigrants for changing optimization problems, *in* 'Applications of Evolutionary Computing', Vol. 4448 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 627–636.

References III

Yang, S. (2008), 'Genetic algorithms with memory- and elitism-based immigrants in dynamic environments', *Evolutionary Computation* **16**(3), 385–416.

Yang, S., Jiang, Y. & Nguyen, T. (2013), 'Metaheuristics for dynamic combinatorial optimization problems', *IMA Journal of Management Mathematics* **24**(4), 451–480.

Yang, S. & Yao, X. (2005), 'Experimental study on population-based incremental learning algorithms for dynamic optimization problems', *Soft Computing* **9**(11), 815–834.

Yazdani, D., Cheng, R., Yazdani, D., Branke, J., Jin, Y. & Yao, X. (2021), 'A survey of evolutionary continuous dynamic optimization over two decades—part a', *IEEE Transactions on Evolutionary Computation* **25**(4), 609–629.

Yazdani, D., Nasiri, B., Sepas-Moghaddam, A. & Meybodi, M. R. (2013), 'A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization', *Applied Soft Computing* **13**(4), 2144–2158.

Yazdani, D., Omidvar, M. N., Cheng, R., Branke, J., Nguyen, T. T. & Yao, X. (2022), 'Benchmarking continuous dynamic optimization: Survey and generalized test suite', *IEEE Transactions on Cybernetics* **52**(5), 3380–3393.

Yu, X., Tang, K., Chen, T. & Yao, X. (2009), 'Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization', *Memetic Computing* **1**(1), 3–24.