Towards a More Practically Sound Formulation of Dynamic Problems and Performance Evaluation of Dynamic Search Methods

Ali Ahrari
School of Engineering and IT
University of New South Wales
Canberra, Australia
a.ahrari@unsw.edu.au

Saber Elsayed
School of Engineering and IT
University of New South Wales
Canberra, Australia
s.elsayed@unsw.edu.au

Ruhul Sarker
School of Engineering and IT
University of New South Wales
Canberra, Australia
r.sarker@unsw.edu.au

Daryl Essam
School of Engineering and IT
University of New South Wales
Canberra, Australia
d.essam@unsw.edu.au

Carlos A. Coello Coello
Departamento de Computación
CINVESTAV-IPN
Mexico City, Mexico
ccoello@cs.cinvestav.mx

***Abstract*—The commonly used methodology for the simulation of dynamic problems formulates them as intervals of static problems, in which the change occurs between two successive intervals. This study proposes a more practically sound formulation of steadily changing dynamic problems, a class of dynamic problems in which the problem landscape continuously, but smoothly, changes over time. The new formulation provides more flexibility for a dynamic optimizer to choose the trade-off between the change frequency and the change severity while the change rate is prescribed by the actual problem. Besides, this study introduces a novel performance indicator for dynamic optimization methods. Unlike conventional ones, this indicator considers the real-time change in the actual problem during a time step and the period in which the best solution should be implemented. The practical importance of this formulation and the proposed performance indicator are studied on a few carefully designed controlled experiments. Subsequently, more comprehensive numerical simulations are performed to investigate the dependency of the optimal change frequency on the employed prediction method and test problem.**

***Index Terms*—Dynamic Problem, Problem Formulation, Performance indicator, Evolutionary algorithm, Steadily changing environment**

I. Introduction

Dynamic optimization deals with finding and tracking the optimal solutions to problems that change over time. For example, the optimal solution to a vehicle routing problem may change because of a change in traffic conditions, road accidents, arrival/cancellation of new orders and so on. Dynamic optimization has been successfully applied to several classes of real-world problems such as dynamic vehicle routing [1] and optimal control of time-varying systems [2].

In some applications, the change in a dynamic problem may be occasional, unexpected, abrupt, chaotic, and/or significant. The most straightforward methodology to deal with such problems is to reinitialize a static optimization method from scratch whenever a change has occurred. However, this methodology can be a reasonable choice only if there is sufficient time or budget to re-optimize the problem, or when the changes are so severe that a change results in a totally new problem which hardly resembles the old one. In many applications, these changes are not radical [3]. Therefore, a more efficient methodology should utilize information from the optimization history, e.g. the history of optimal solutions(s) in the past.

Several strategies have been recently developed to handle the dynamic nature of these problems. Some examples are memory-based methods [4] prediction methods [5], diversity-based methods [6], and multi-population methods [7]. These strategies are generally incorporated into a static optimizer to form a dynamic optimization method. In particular, some of these strategies, like prediction strategies, are only activated immediately after a change and remain inactive until the next change. If changes are not informed, a dynamic optimizer needs a change detection mechanism as well. This mechanism is usually based on re-evaluation of a fraction of population members or analyzing the behavior of the optimization process [4].

There is one interesting class of dynamic problems, which has been referred to as steadily changing problems [8]. For these problems, there is no distinguishable change, but changes occur continuously over time. For example, the load on the electricity network steadily but smoothly changes with daylight time. Some other examples are supplying fuel to patrolling boats by a ship, or delivering hazardous materials to multiple moving units by a robot [9]. The conventional method to formulate such dynamic problems is to discretize the problem to intervals of size τ_t, which is known as

978-1-7281-2547-3/20/$31.00 ©2020 IEEE

the change frequency. The problem is assumed to remain unchanged during each interval [10]. The change frequency can be small, resulting in more frequently but less severely changing problems (more accurate representation of the actual problem) [8]. In contrast, a greater τ_t means less frequent but more severe changes in the formulated problem. The value of τ_t is thus a part of the problem formulation. As discussed, the actual problem may be steadily changing, and selection of τ_t should not be a part of the dynamic problem. Instead, this parameter should be regarded as a setting for the optimization method. Certain dynamic optimization methods may work better with less frequent changes (but more severe ones), whereas some other methods may be more efficient when the changes are less severe (but occur more frequently).

Another important aspect of dynamic optimization is how the performance indicator is calculated. The commonly accepted methodology is to average a static performance indicator at the end of each time step (IGD, IGD$^+$, or hypervolume for multi-objective problems [8], [11], [12]). However, such an indicator may not accurately predict the performance of a solution when applied to the actual problem. This issue has been analyzed by Deb et al. [8], and is briefly explained here.

The actual problem is captured and formulated as an optimization problem at time $t = 0$. A static optimization is then performed and a (near-) optimal solution $\boldsymbol{x}_{\text{best}}$ is found. The optimization process, however, has taken a time of τ_t. Therefore, $\boldsymbol{x}_{\text{best}}$ is available at time τ_t, from which this solution can be applied to the actual problem. At this time, $\boldsymbol{x}_{\text{best}}$ might not be a near-optimal solution for the actual problem anymore. On the other hand, this solution should be applied to the actual problem for an interval, known as the *implementing window* [8], during which the optimum of the actual problem further deviates from $\boldsymbol{x}_{\text{best}}$. A practically sound performance indicator should take such changes in the actual problem into account when quantifying the performance of a dynamic search method.

This study intends to improve existing methodologies for the formulation of a dynamic problem and performance quantification of dynamic optimization methods. In summary:

- It develops a more practically sound formulation of steadily changing dynamic problems in which the change frequency is no longer a problem setting but a part of algorithmic settings.
- It shows that, for a fixed change rate, the optimal change frequency is both problem-dependent and algorithm-dependent.
- A new performance indicator is developed which, unlike conventional indicators, takes real-time changes in the actual problem during the optimization process and the implementing window into account.
- It analyzes the difference between the proposed performance indicator and the conventionally adopted ones for different dynamic multi-objective optimization (DMO) methods.

In this study, the dynamic problem is assumed to be a multi-objective problem. However, the findings can be easily reduced to single objective dynamic problems. The rest of this study is organized as follows: Section II reviews the related work on the formulation of dynamic problems and existing performance indicators for dynamic optimization. Section III develops an alternative formulation for dynamic problems and a novel performance indicator for dynamic methods. Section IV performs two descriptive experiments to illustrate the importance of the new formulation and trade-off between change frequency and change severity. Section V investigates this trade-off for different prediction methods and test problems. Finally, conclusions are drawn in Section VI.

II. Related work

The type of dynamic problems considered in this study can be formulated as follows:

$$\begin{aligned} &\text{Min.} \quad F(\boldsymbol{x},t) = [f_1(\boldsymbol{x},t),\ f_2(\boldsymbol{x},t),\ \ldots, f_M(\boldsymbol{x},t)]^T \\ &\text{s.t.} \quad \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U \end{aligned} \tag{1}$$

In this formulation, F is the vector of objective functions, M is the number of objectives, and $\boldsymbol{x}^L$ and $\boldsymbol{x}^U$ define the lower and upper range of the search space, respectively. This equation means the search range is fixed, but the relationship between the decision variables and the objective functions change over time.

The commonly used formulation of dynamic problems assumes that the problem remains unchanged during intervals of τ_t iterations. Each interval is called a time step, which is calculated as follows:

$$t = \max\left\{0,\ \left\lceil \frac{i - T_0}{\tau_t} \right\rceil - 1\right\}, \tag{2}$$

in which t is the index of the time step and i is the iteration number. Parameter T_0 specifies when the first change occurs. The length of the first time step might be set to a different value than τ_t. In this study, the iteration number is also used to measure the time since the start of the optimization process. This time should not be confused with the time step.

Fig. 1 illustrates this formulation for a typical dynamic problem. The optimization process starts from a randomly generated population $\mathbf{X}_{\text{IP}}^{(t=0)}$. Then, a static optimization process is performed for a period of τ_t iterations. After that, a change occurs in the problem, the severity of which is defined by the change severity parameter ($n_t > 0$). A greater n_t means the problem changes less severely after each time step. The population in the final iteration of this time step ($\mathbf{X}_{\text{FP}}^{(t=0)}$) represents the solution(s) provided by the dynamic optimizer for time step zero.

When the first change is detected, the updated problem is provided for the optimization method. The dynamic optimizer employs some information from the history of the optimization process, mainly $\mathbf{X}_{\text{FP}}^{(t=0)}$, to determine the seed population for time step 1 ($\mathbf{X}_{\text{IP}}^{(t=1)}$). A good seed population will make the future search more efficient. The problem at time step 1 is then optimized by the same optimizer. This process continues until a predefined number of changes occur.

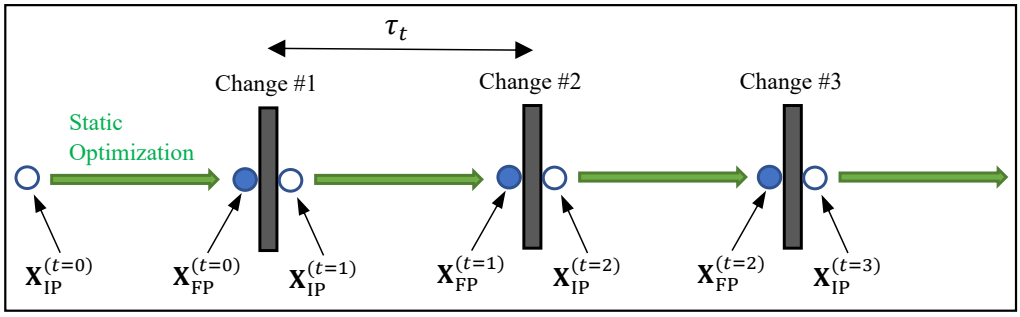


Fig. 1: Commonly used formulation of dynamic problems

The common methodology, such as the one employed in the CEC'2018 test suite for DMO [13] treats τ_t and n_t as a part of the problem settings, e.g. $\tau_t = 30$ and $n_t = 10$. Besides, the performance of a DMO method is commonly calculated by averaging its performance at the end of each time step. For this purpose, first, a well-known static multi-objective performance indicator, such as hypervolume (HV), is calculated at the end of each time step:

$$HVR\,(t) = \frac{HV\left(F\left(\mathbf{X}_{\mathrm{FP}}^{(t)}, t\right)\right)}{HV\left(F\left(\mathbf{S}^{(t)}, t\right)\right)}, \tag{3}$$

in which $\mathbf{S}^{(t)}$ is the true efficient set of the dynamic problem at time step t. $HVR\,(t)$ is regarded as the hypervolume ratio for time step t. Mean HVR is then calculated by averaging the calculated hypervolume ratios at the end of each time step:

$$MHVR = \frac{1}{t_{\max}} \sum_{t=1}^{t=t_{\max}} HVR\,(t) \tag{4}$$

This commonly used methodology provides a simple approach for the formulation of dynamic problems and performance evaluation of a dynamic optimizer. However, this formulation might overlook some practical (and important) features of dynamic problems that steadily change. These features have been highlighted by Deb et al. [8]:

- The actual dynamic problem changes during a time step, whereas the formulated problem does not.
- A selected solution from the population should be applied to the actual problem for an interval (implementing window). During this period, the actual problem is changing while the implemented solution is fixed.

Deb et al. [8] also stated that steadily changing dynamic problems can be simulated as a dynamic problem with frequent but non-severe changes (a small τ_t but a large n_t), or less frequent but more severe changes (a large τ_t but a small n_t). They also considered the former option closer to practice since the effect of dynamic changes during the optimization process is smaller. This means that the value of $\tau_t n_t$, which is called the *change rate* in this study, is prescribed by an actual problem while the selection of τ_t or n_t (but not both) can be left to the employed optimization method.

Deb et al. [8] argued that a fraction of the time step should be used for optimization, and the remaining time should be allocated for implementing the selected solution from the optimization outcome. During both periods, the actual problem is changing. They also theorized that there should be a trade-off between the advantages of using a greater τ_t versus a greater n_t for a predefined $n_t\tau_t$. A greater τ_t provides more time for the optimization method to converge to the optimal solution(s); however, the formulated problem will deviate more from the actual one. They investigated the effect of τ_t for a fixed change rate for FDA2 using dynamic NSGA-II; however, a greater τ_t turned out to be always beneficial. The following reasons may explain the absence of trade-off in their simulation:

- Their employed performance indicator, MHVR, was calculated according to the existing conventional method (see 4)). This means that the change in the actual problem during a time step was ignored.
- Dynamic NSGA-II employs hypermutation for handling dynamic aspects of the problem, which does not make use of the pattern in the changes. Therefore, the presence of simple patterns, which is the case for less severe changes, does not improve the performance of dynamic NSGA-II.

III. Proposed Formulation

Although Deb et al. [8] highlighted the importance of considering the change during a time step and implementing window, their numerical simulation did not involve such factors. Furthermore, they assumed that the optimization process halts during the implementing window. A more practically sound formulation and performance indicator is proposed in this section to address these shortcomings.

A. Novel Formulation

The main features of the proposed formulation in this study are as follows:

- It is possible to continue the dynamic optimization process in the background while the selected solution from previous time steps is being implemented during the implementing window.
- The length of the implementing window (τ_w) and the change rate are prescribed by the dynamic problem.

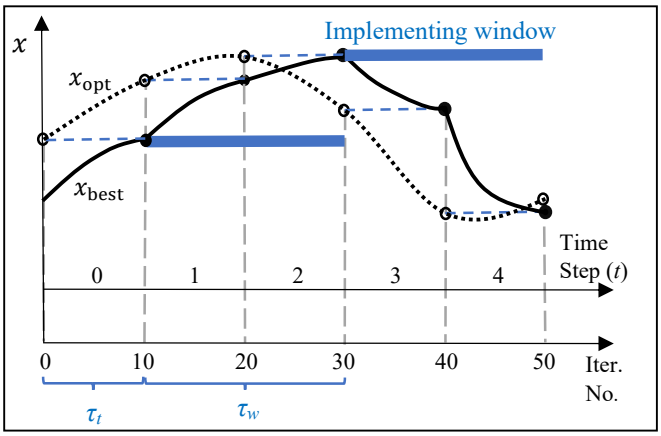


Fig. 2: Illustration of the simulation of a dynamic problem with one decision parameter. The solid line represents the best solution value, whereas the dotted line represents the actual optimum of the dynamic problem.

However, the selection of τ_t is left for the optimization method.

An exemplary case with one objective and one decision parameter is illustrated in Fig. 2 for $\tau_t = 10$ and $\tau_w = 20$. The problem initiates at time 0 when the actual problem is captured and formulated. This formulated problem remains unchanged for a period of τ_t iterations. The optimization process then initiates from a random population and continues until the tenth iteration. During this period, the best solution found by the search method gets closer to the optimal solution of the formulated problem at time 0, which is $x_{\text{opt}}^{(t=0)}$. However, since the optimization process takes time, the actual problem, and thus, the optimal solution of the actual problem changes while the formulated problem is being optimized. Therefore, after τ_t iterations, $x_{\text{best}}^{(t=1)}$ approximates $x_{\text{opt}}^{(t=0)}$, not $x_{\text{opt}}^{(t=1)}$. Furthermore, this best-found solution is expected to be implemented for τ_w iterations, during which the actual problem changes even more. For this example, the implementing period is 20 iterations ($\tau_w = 20$). Although the implemented solution does not change during the implementing window, the optimization process can continue in the background. For example, the formulation of the actual dynamic problem can be updated after each τ_t iterations. From iteration 11 to iteration 20, the optimization process gradually converges to $x_{\text{opt}}^{(t=1)}$. This solution cannot be implemented but it may be useful to speed up convergence in subsequent time steps.

B. Performance Evaluation

The proposed performance indicator in this study aims to address the drawback of the conventionally used one. More specifically, it intends to consider the actual fitness of the obtained solution(s) during the implementing window instead of their fitness at the end of each time step of the formulated problem. For this study, $T_0 = 0$, which means the problem is steadily changing from the first iteration. Although it is not mandatory, for benchmarking purposes, it is assumed that only solution(s) at the end of each time step are recorded and may be used for implementation.

First, the whole optimization process is divided into N_w equally sized implementing windows. For the implementing window that starts at iteration i, the latest solutions that can be applied are those at the end of time step t, in which t is calculated as follows:

$$t = \frac{\left(\left\lceil \frac{i}{\tau_w} \right\rceil - 1\right)\tau_w}{\tau_t} - 1 \tag{5}$$

When a solution from $\mathbf{X}_{\text{FP}}^{(t)}$ is implemented at time i, its actual fitness is the one calculated according to the actual problem (which steadily changes) at this time. It is worth mentioning that the only difference between the actual and the formulated problem is that the actual problem changes steadily while both problems have identical change rate. The formulated problem with a very small τ_t can reliably represent a steadily changing problem. Therefore, this study considers the formulated problem when $\tau_t = 1$ and an identical change rate as an equivalent to the actual problem. The objective function of this equivalent problem is denoted by F_e. The real-time hypervolume ratio (RTHVR) at iteration i is then calculated as follows:

$$\mathit{RTHVR}\,(i) = \frac{\mathit{HV}\left(F_e\left(\mathbf{X}_{\text{FP}}^{(t)}, i\right)\right)}{\mathit{HV}\left(F_e\left(\mathbf{S_e}^{(i)}, i\right)\right)}, \tag{6}$$

in which $\mathbf{S_e}^{(i)}$ is the efficient set of the equivalent problem at iteration i. The overall performance is then calculated by averaging $\mathit{RTHVR}\,(i)$ over the optimization process:

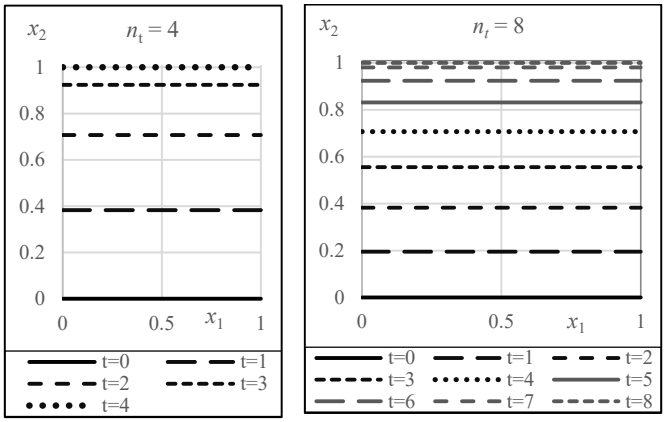
$$\mathit{MRTHVR} = \frac{1}{i_{\max} - \tau_w} \sum_{i=\tau_w+1}^{i_{\max}} \mathit{MHVR}\,(i), \tag{7}$$

in which $i_{\max}$ is the maximum number of iterations. The first τ_w iterations were excluded from the calculation of *MRTHVR*.

IV. Descriptive Experiments

This section performs two descriptive experiments to provide insights into the proposed formulation and performance indicator. For this purpose, a simple initialization method is used in combination with a slightly modified NSGA-III [14] to form a DMO method. This modification employs a simpler heuristic to estimate the Nadir point: the union of the parent and offspring population is sorted according to their non-domination rank. The Nadir point is the maximum of all objective values of population members whose non-domination rank is equal or less than the critical rank. The critical rank is the rank of the N^{th} solution (N is the population size). This modification was motivated by a theoretical issue regarding the hyperplane-based heuristic in NSGA-III for estimating the Nadir point [15]. Other parameters of the modified NSGA-III are set following the recommended settings as follows:

- $N = 100$.



(a) (b)

Fig. 3: The efficient set of DF1 in different time steps when a) $n_t = 4$ and b) $n_t = 8$

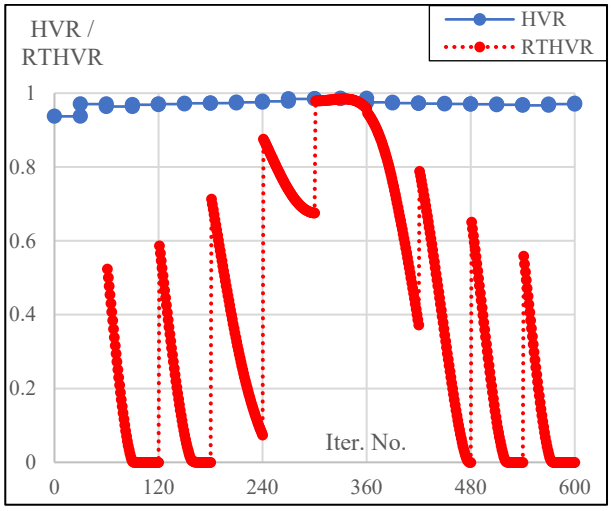


Fig. 4: Performance of the developed DMMO method: HVR versus DHVR for DF1 when $\tau_t = 30$, $n_t = 10$, $\tau_w = 60$. Note that each 30 iterations represent one time step.

- Simulated Binary Crossover (SBX): $P_{\text{cross}} = 0.9$, swap probability of 0.5, η_c = 20
- Polynomial-based mutation: P_{mut}=1/D, η_m = 20;

in which D is the problem's dimensionality.

A simple prediction method, called controlled random variation (CRV), is used in this section as the prediction method. This method adds some random variation to the final population of the previous time step (time step t) to generate the initial population for the new time step (time step $t+1$):

$$\boldsymbol{x}_{\text{IP}_j}^{(t+1)} = \boldsymbol{x}_{\text{FP}_j}^{(t)} + \mathcal{N}_D\left(\mathbf{0}, \sigma_{\text{r}}^2\right), \; j = 1, 2, \ldots, N \tag{8}$$

in which $\mathcal{N}_D\left(0, \sigma_{\text{r}}^2\right)$ is a vector of D randomly generated numbers from the normal distribution with mean $\mathbf{0}$ and standard deviation σ_{r}, which is calculated as follows:

$$\sigma_{\text{r}} = \frac{\left\|\boldsymbol{c}^{(t)} - \boldsymbol{c}^{(t-1)}\right\|}{2\sqrt{D}}, \tag{9}$$

in which $\boldsymbol{c}^{(t)}$ is the centroid of $\mathbf{X}_{\text{FP}}^{(t)}$. For the first change, $\sigma_{\text{r}} = 0$.

The employed test problem in this section is DF1 [13]. In this two-objective function, (Fig. 3), the efficient set moves along a fixed direction over time. The length of this movement is controlled by n_t. Therefore, it is an excellent example to study the effect of τ_t for a fixed change rate.

A. *HVR versus RTHVR*

To show the difference between HVR and RTHVR, DF1 is optimized when τ_t= 30, n_t= 10, and τ_w= 60. Optimization continues for 20 Changes. Fig. 4 illustrates the calculated values of HVR (at the end of each time step) and RTHVR (for each iteration after the 60^{th} iteration). It can observed that:

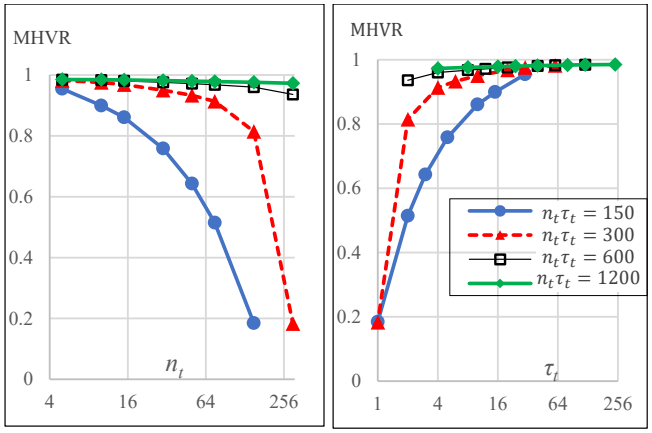
- RTHVR is generally (but not always) maximal at the beginning of each time step (iteration numbers 61, 121, 181, ..., 540). However, even at this specific iteration, RTHVR $\leq$ HVR. The reason for this is that RTHVR considers the change in the actual problem during a time step.
- HVR is calculated at the end of each time step, whereas RTHVR changes during each time step. This change is generally a reduction in the performance indicator.
- For this function, the gap between HVR and RTHVR is huge at the early time steps because the change in the efficient set of two successive time steps is huge. Therefore, RTHVR is much less than HVR, even at the beginning of each implementing window. Furthermore, RTHVR reduces fast during this window. In contrast, the difference between the efficient sets of the problem in time steps 9-11 is small, resulting in a smaller gap between HVR and RTHVR, as well as a lower reduction rate in RTHVR over the corresponding implementing window.

This experiment has shown that there can be a huge gap between the real-time performance of the implemented solution (RTHVR in this case) and the one calculated according to the existing performance indicator (HVR in this case).

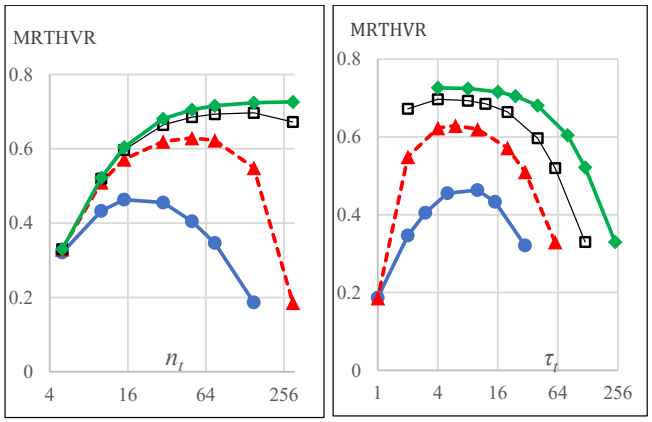
B. *Effect of the Change Severity*

This experiment compares the effect of the change severity for a fixed change rate on MHVR and MRTHVR. For this experiment, $\tau_w = 0.2\tau_t n_t$ and the maximum number of changes is set to $4n_t$. This means that the cumulative changes in the problem and during the implementing window are identical for all values of τ_t and n_t. Fig. 5 shows MHVR and MRTHVR as a function of the change severity (n_t) for selected values of the change rate ($\tau_t n_t$). This figure reveals that:

- For a fixed $\tau_t n_t$, a smaller n_t (or a greater τ_t) always improves MHVR. This observation parallels the results



(a) MHVR as a function of n_t and τ_t



(b) MRTHVR as a function of n_t and τ_t

Fig. 5: Effect of the change severity (n_t) and change rate ($n_t\tau_t$) on the calculated performance indicators.

reported in [8] for FDA2. This trend is more severe for a smaller $\tau_t n_t$.

- In contrast, an optimal trade-off between the advantages and disadvantages of a greater τ_t can be observed when MRTHVR is employed as the performance indicator.
- When MRTHVR is considered, the optimal value of n_t substantially increases when $n_t\tau_t$ is greater. In contrast, the optimal value of τ_t does not considerably change.

V. Numerical Comparison

The results of descriptive experiments revealed the fundamental difference between the proposed and existing performance indicators for DMO for a simple prediction method and a simple DMO test problem. It also revealed the presence of an optimal value for n_t. This section investigates these effects for a more diverse set of test problems and prediction methods.

The employed test problems in this section are a subset of the CEC'2018 test suite for DMO [13]. These problems were selected with an emphasis on the smoothness of the change and the diversity in the change pattern of the efficient set. These test problems are DF1, DF3, DF4, DF7, DF10, DF12. The first four problems have two objectives, whereas the last two problems have three objectives.

Furthermore, six prediction methods are considered in combination with the modified NSGA-III used in the previous section. These prediction methods are: Controlled Translation with Random and Directional Variation (CTRDV) [16], Controlled Random Variation (CRV), the prediction method from Steady State and Generational Evolutionary Algorithm [17], which is denoted by SSGEA, the pointwise prediction method (PRE) [18], Multidirectional Prediction (MDP) approach [11], and the hypermutation (HM) method developed in [8].

For all the problems, $\tau_t n_t = 240$; however, ten different values for n_t are used to formulate the problem. These values range from 4 to 240. A maximum of $5n_t$ changes are considered and $\tau_w = 60$. Each prediction method is used to solve each formulated problem 20 times independently to calculate MHVR and MRTHVR. Fig. 6 illustrates MHVR and MRTHVR obtained from each method, each problem, and each value of n_t. The value of n_t^* for each method and each problem is provided in Fig. 7.

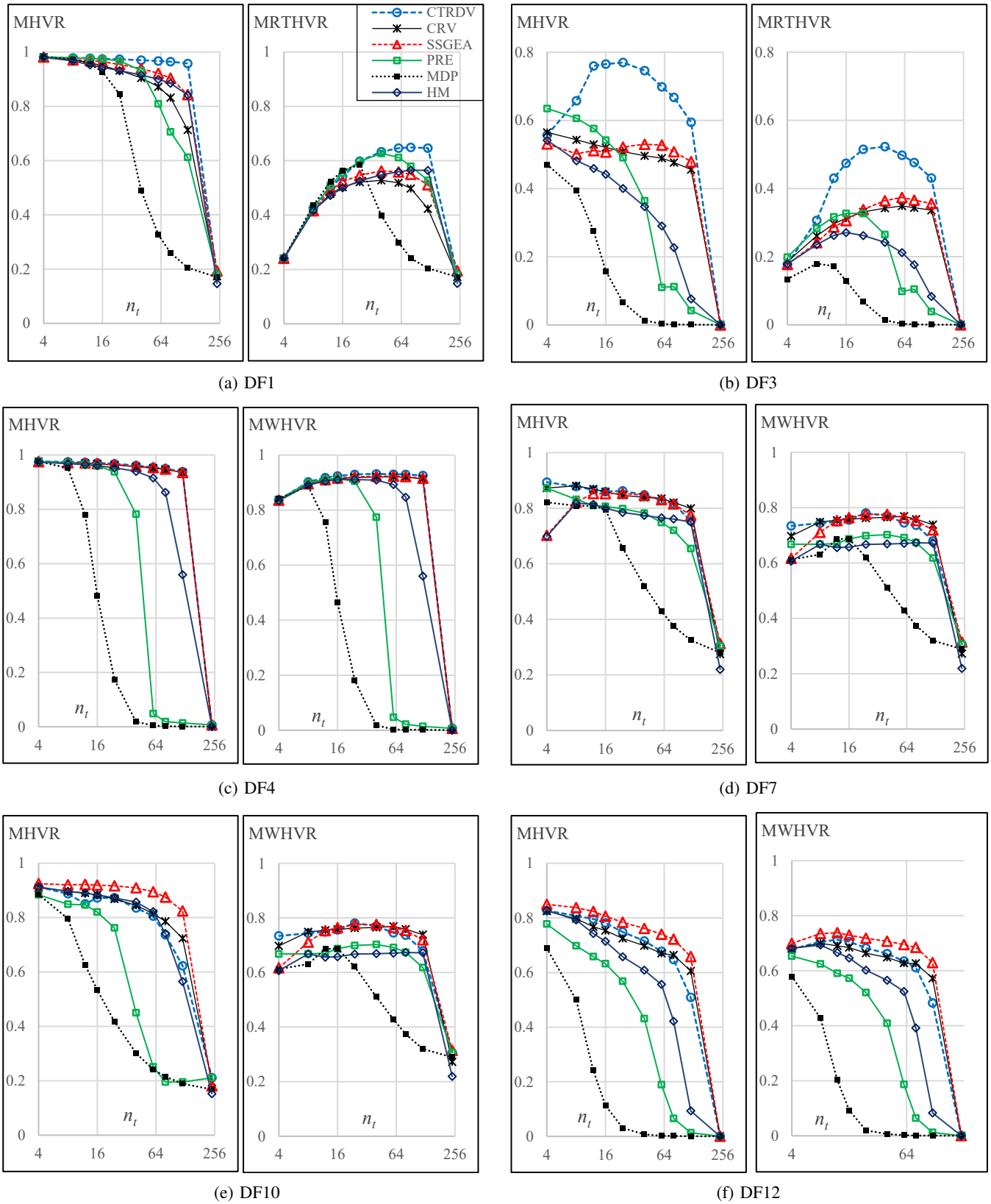
Fig. 6 shows that for DF1, DF4, DF10 and DF12, MHVR of all methods improves when n_t decreases. For DF3 and DF7, this trend is generally observable except for a few prediction methods. In contrast, when MRTHVR is the performance indicator, most prediction methods show an optimal value for n_t which is not the minimum or the maximum of the tested values.

When MRTHVR is the performance indicator, the optimal value of n_t strongly depends on the prediction method (Fig. 6). For example, for DF3, $n_t^* = 8, 16, 40, 60$ for MDP, PRE, CTRDV, and SSGEA, respectively. The optimal n_t depends on the problem as well. For example, for CTRDV, $n_t^* = 4, 24, 40, 80$ for DF10, DF7, DF3, and DF1, respectively.

Fig. 7 shows that when MRTHVR is the performance indicator, there is hardly a detectable trend in relative values of n_t^* for different prediction methods. For example, for DF3, n_t^* for CTRDV is much higher than n_t^* for HM, while the situation is the opposite for DF7. The only exception is MDP, which has the smallest n_t^* almost for all problems.

VI. Summary and Conclusions

In steadily changing dynamic problems, the problem landscape continuously, but smoothly changes over time. The common methodology for the simulation of dynamic problems formulates these problems as piecewise static ones, in which the change occurs at the end of each time step. This study has proposed a more practically sound formulation for steadily changing dynamic problems. In this formulation, the change rate is prescribed by the problem. However, selection of the change frequency or change severity (but not both) is a matter of choice which is left to the employed optimization method. The numerical simulation in this study revealed that there is generally an optimal value with the best trade-off between the advantages of less frequent changes and less severe ones. Nevertheless, this optimal value is both problem-dependent and method-dependent.



(a) DF1

(b) DF3

(c) DF4

(d) DF7

(e) DF10

(f) DF12

Fig. 6: Calculated performance indicators (MHVR and MWHVR) as a function of n_t for the tested problems

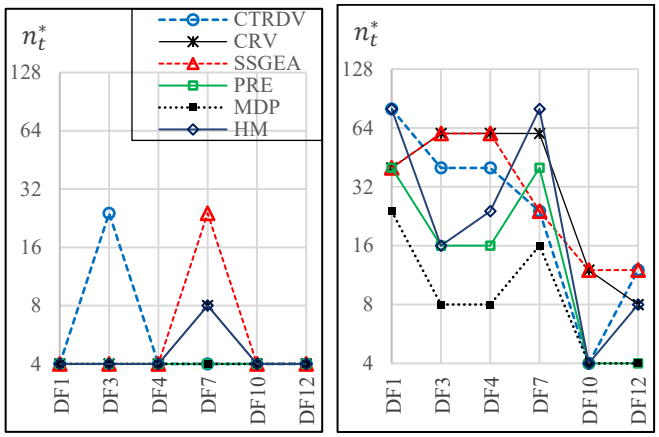


Fig. 7: The optimal value for the change severity (n_n^*) for different prediction methods and different problems when the performance indicator is a) MHVR and b) MRTHVR

The performance of a dynamic optimization method is commonly measured by averaging its performance at the end of each time step. This performance is calculated by comparing the fitness of the solutions at the end of each time step and the optimal solution(s) for the formulated problem for that time step. However, such indicators ignore the change in the actual problem during the time step. Besides, in practice, the selected solution from the optimization process should be applied for a finite time (implementing window), during which the applied solution may not change. During this period, the actual problem further deviates from the formulated problem from the optimization of which the implemented solution was selected.

The proposed performance indicator in this study, called real-time hypervolume ratio (RTHVR), addresses these shortcomings by considering the real-time change in the actual problem during a time step and the implementing window. It provides a more practically meaningful measure that can predict the average performance of the implemented solution during the implementing window. For a fixed change rate, the presence of an optimal trade-off value for the change frequency has been observed when mean RTHVR was employed as the performance indicator. This performance indicator also highlights the importance of finding solutions which are not only near-optimal but also insensitive to small changes in the problem.

So far, the potential pattern in the changes of a dynamic problem has been mainly used for predicting the location of the new optimal solutions whenever a change occurs. The findings in this study open new possibilities for using such a pattern: how good the optimized solution will be during the implementing window? This consideration can help the optimization process find solutions that are predicted to perform best during the implementing window, which is ultimately, the goal of dynamic optimization.

VII. Acknowledgments

The research is partly supported by a Australian Research Council project ARC DP190102637. Computational work in this project was supported by Australian National Computational Infrastructure. The last author acknowledges support from CONACyT grant no. 1920 and from a SEP-Cinvestav grant (application no. 4).

References

[1] N. R. Sabar, A. Bhaskar, E. Chung, A. Turky, and A. Song, "A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion," *Swarm and evolutionary computation*, vol. 44, pp. 1018–1027, 2019.

[2] S. Scolan, S. Serra, S. Sochard, P. Delmas, and J.-M. Reneaume, "Dynamic optimization of the operation of a solar thermal plant," *Solar Energy*, vol. 198, pp. 643–657, 2020.

[3] L. Cao, L. Xu, E. D. Goodman, and H. Li, "Decomposition-based evolutionary dynamic multiobjective optimization using a difference model," *Applied Soft Computing*, vol. 76, pp. 473–490, 2019.

[4] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[5] A. Meier and O. Kramer, "Prediction in nature-inspired dynamic optimization," in *Frontier Applications of Nature Inspired Computation*. Springer, 2020, pp. 34–52.

[6] G. Ruan, G. Yu, J. Zheng, J. Zou, and S. Yang, "The effect of diversity maintenance on prediction in dynamic multi-objective optimization," *Applied Soft Computing*, vol. 58, pp. 631–647, 2017.

[7] R. Vafashoar and M. R. Meybodi, "A multi-population differential evolution algorithm based on cellular learning automata and evolutionary context information for optimization in dynamic environments," *Applied Soft Computing*, vol. 88, p. 106009, 2020.

[8] K. Deb, S. Karthik *et al.*, "Dynamic multi-objective optimization and decision-making using modified nsga-ii: a case study on hydro-thermal power scheduling," in *International conference on evolutionary multi-criterion optimization*. Springer, 2007, pp. 803–817.

[9] Q. Jiang, R. Sarker, and H. Abbass, "Tracking moving targets and the non-stationary traveling salesman problem," *Complexity International*, vol. 11, no. 2005, pp. 171–179, 2005.

[10] W. T. Koo, C. K. Goh, and K. C. Tan, "A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment," *Memetic Computing*, vol. 2, no. 2, pp. 87–110, 2010.

[11] M. Rong, D. Gong, Y. Zhang, Y. Jin, and W. Pedrycz, "Multidirectional prediction approach for dynamic multiobjective optimization problems," *IEEE transactions on cybernetics*, vol. 49, no. 9, pp. 3362–3374, 2018.

[12] A. Ahrari, S. Elsayed, R. Sarker, D. Essam, and C. A. C. Coello, "Weighted pointwise prediction method for dynamic multiobjective optimization," *Information Sciences*, pp. in press, doi: 0.1016/j.ins.2020.08.015, 2020.

[13] S. Jiang, S. Yang, X. Yao, K. C. Tan, M. Kaiser, and N. Krasnogor, "Benchmark problems for cec2018 competition on dynamic multiobjective optimisation," Newcastle University, Tech. Rep., 2017.

[14] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[15] J. Blank, K. Deb, and P. C. Roy, "Investigating the normalization procedure of nsga-iii," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2019, pp. 229–240.

[16] A. Ahrari, S. Elsayed, R. Sarker, and D. Essam, "A new prediction approach for dynamic multiobjective optimization," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 2268–2275.

[17] S. Jiang and S. Yang, "A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 65–82, 2017.

[18] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2007, pp. 832–846.