

# Dynamic Time-linkage Problems Revisited

Trung Thanh Nguyen<sup>1</sup> and Xin Yao<sup>1</sup>

The Centre of Excellence for Research in Computational Intelligence and Applications  
(CERCIA), School of Computer Science,  
University of Birmingham, B15 2TT, United Kingdom  
{T.T.Nguyen, X.Yao}@cs.bham.ac.uk

**Abstract.** Dynamic time-linkage problems (DTPs) are common types of dynamic optimization problems where "decisions that are made now ... may influence the maximum score that can be obtained in the future"[3]. This paper contributes to understanding the questions of what are the unknown characteristic of DTPs and how to characterize DTPs. Firstly, based on existing definitions we will introduce a more detailed definition to help characterize DTPs. Secondly, although it is believed that DTPs can be solved to optimality with a perfect prediction method to predict function values [3] [4], in this paper will discuss a new class of DTPs where it might not be possible to solve the time-linkage problems to optimality because there is not always the possibility to perfectly predict the future. In addition, in this type of DTPs if we try to predict the future based on information from the past, we may even get worse results than not using a predictor at all. We will also propose a benchmark problem to study that particular type of time-linkage problems.

## 1 Introduction

This paper studies some unknown characteristics and the solvability of some classes of dynamic time-linkage problems (DTP)<sup>1</sup>. DTP is a common type of dynamic optimization problems (DOPs) in both real-world combinatorial (scheduling [6], vehicle routing [8], inventory management [4]) and continuous domains (non-linear predictive control [13], optimal control theory[7]) but has not yet received enough attention from the Evolutionary Algorithms research. They are defined as problems where "...there exists at least one time  $0 \leq t \leq t^{end}$  for which the dynamic optimization value at time  $t$  is dependent on at least one earlier solution..." [5].

Although the importance of DTPs have been shown through their presence in a broad range of real-world applications, due to the lack of research attention there are still many characteristics that we do not fully know about this type of problems. For example, how should we define and classify DTPs in detail; is

---

<sup>1</sup> "Dynamic time-linkage problem" is the term given by Bosman[3]. In other fields this type of problems is given different names, for example "model predictive control" or "anticipative decision process" although the nature of the problems is the same.

there any characteristics of DTPs that we do not know; with these characteristics are DTPs still solvable; and what is the appropriate strategy to solve them.

This paper contributes to the work on finding partial answers for the above questions. Firstly, based on existing definitions we will introduce a more detailed definition to help characterize DTPs and other DOPs. Secondly, although it is believed that DTPs can be solved to optimality with a perfect prediction method to predict future function values [3] [4], in this paper will discuss a new class of DTPs where it might not be possible to solve the time-linkage problems to optimality because there is not always the possibility to perfectly predict the future. In addition, in this type of DTPs if we try to predict the future based on information from the past, we may even get worse results than not using a predictor at all. We will also propose a benchmark problem to study that particular type of DTPs.

## 2 Existing definition of dynamic time-linkage problems

A DTP is firstly a dynamic optimization problem, hence it also has all the characteristics of a regular DOP. The additional feature of a DTP, which makes it different from normal DOPs, is that the dynamic of a parameter may depend not only on the time variable, but also on earlier decisions made by the algorithm. It means that at the current time  $t^{now}$  the value of the parameter  $\gamma(t^{now})$  of a function  $f$  may depend on the value of the variable  $x(t)$ ,  $0 \leq t < t^{now}$  found by the algorithm at at least one point before  $t^{now}$ .

Equation 1 shows the formal definition (proposed in [3] ) for DOPs (including DTPs) with the time variable  $t \in \mathbb{T} = [0, t^{end}]$ ,  $t^{end} > 0$ .

$$\begin{aligned} & \max \{F_\gamma(x(t))\} \text{ subject to } C_\gamma(x(t)) = \textit{feasible} \text{ with} \\ & F_\gamma(x(t)) = \int_0^{t^{end}} f_{\gamma(t)}(x(t)) dt \\ & C_\gamma(x(t)) = \begin{cases} \textit{feasible} & \text{if } \forall t \in [0, t^{end}] : C_{\gamma(t)}(x(t)) = \textit{feasible} \\ \textit{infeasible} & \text{otherwise} \end{cases} \end{aligned} \quad (1)$$

where  $\gamma$  are the time-dependent parameters of  $f$  and  $C$  is the constraint function.

Although the definition above is useful to generalize DTPs, it might not be detailed enough if we want to characterize one important property of DTP instances: *algorithm-dependency*. We consider DTP instances algorithm-dependent because the structure of a DTP in the future may depend on the current value of  $x(t)$ , which in turn depends on the algorithm used to solve the problem. At a particular change step  $t$ , different algorithms might provide different solutions  $x(t)$ , hence changing the future problem in different ways. Because of this property, we believe that in order to define a DTP in an unambiguous way, the algorithm used to solve a problem instance should be considered a part of that instance. The original definition in eq. 1 does not fully encapsulate this.

Another reason for us to formulate an extended definition is that in (eq. 1) the time-linkage feature is not explicitly expressed. Instead, that feature is encapsulated in the expression of  $f_{\gamma(t)}$ . It would be better if the time-linkage

property can be captured explicitly in the definition. This has been partially done in [5] and here we will extend that concept further by including previous solutions that affect future function values in the definition.

In addition, because a DTP is firstly a dynamic problem, it would be useful if its definition is formulated in a detailed enough level to cover all characteristics of a DOP. Such a detailed definition would help to answer the question of (1) how to classify/characterize DTPs effectively and (2) how to separate the difficulty caused by the underlying static problem from the difficulty caused by the dynamic/time-linkage feature. To provide such a detailed level, the proposed definition will cover such aspects of a DOP as how to control changes, how to express changes in the domain range/constraints, how to control change frequency and how to integrate the frequency of change into problem descriptions. Some aspects as time unit [1], [12], controlling changes [12] [9], and changes in constraints [3], [5] have been mentioned elsewhere. In this paper these aspects will be described in a more formal and detailed level. Other aspects are introduced for the first time in this paper.

### 3 A more detailed definition for DTPs

**Definition 1 (Optimization algorithm and its solutions).** *Given a dynamic problem  $f_t$  at the time step  $t$  and a set  $P_t$  of  $k_t$  point  $\mathbf{x}_1, \dots, \mathbf{x}_{k_t} \in S_t$  where  $S_t \subseteq \mathbb{R}^n$  is the search space<sup>2</sup>, an optimization algorithm  $G$  can be seen as a mapping:  $G_t : \mathbb{R}^{n \times k_t} \rightarrow \mathbb{R}^{n \times k_{t+1}}$  capable of producing a solution set  $P_{t+1}$  of  $k_{t+1}$  optimised points  $\mathbf{x}_1^G, \dots, \mathbf{x}_{k_{t+1}}^G$  at the time step<sup>3</sup>  $t + 1$ :  $P_{t+1} = G_t(P_t)$ .<sup>4</sup> Generally, at a time step  $t^e \in \mathbb{N}^+$  the set of solutions that we get by applying an algorithm  $G$  to solve  $f_t$  with a given initial populations  $P_{t^b-1}$  during the period  $[t^b, t^e]$ ,  $t^b \geq 1$ , is denoted  $X_{f_t}^{G[t^b, t^e]} = \bigcup_{t=t^b}^{t^e} P_t = \bigcup_{t=t^b}^{t^e} G_t(P_{t-1})$*

**Definition 2 (Full-description form).** *Given a mathematical expression  $\widehat{f}_\gamma(x)$  with a variable  $x$  and a set of parameters  $\gamma \in \mathbb{R}^m$ , we call  $\widehat{f}_\gamma(x)$  the full-description form of a set of mathematical expressions  $\{f_1(x), \dots, f_n(x)\}$  if there exists a set of vectors  $\{\mathbf{c}_1, \dots, \mathbf{c}_n\}$ ,  $\mathbf{c}_i \in \mathbb{R}^m, i = 1 : n$  so that if  $\gamma = \mathbf{c}_i$  then  $\widehat{f}_{\gamma=\mathbf{c}_i}(x)$  is equal to  $f_i(x)$ . In other words:*

$$\begin{aligned} \widehat{f}_\gamma(x) &\xrightarrow{\gamma=\mathbf{c}_1} f_1(x) \\ &\dots \\ \widehat{f}_\gamma(x) &\xrightarrow{\gamma=\mathbf{c}_n} f_n(x) \end{aligned} \tag{2}$$

<sup>2</sup> Here we are considering search spaces  $\subseteq \mathbb{R}^n$  but it can be generalized for non-numerical encoding algorithms by replacing  $\mathbb{R}^n$  with the appropriate encoding space.

<sup>3</sup> As mentioned in [1] and [12], from the perspective of optimization algorithms time is discrete and the smallest time unit is one function evaluation.

<sup>4</sup> To some extents the mapping  $G$  is similar to the *generation transition function* (GTF) used to represent the EA population sequence in [2]. However, different from GTF, in this paper  $G$  is used to represent any type of optimization algorithms.

Each function  $f_i(x)$ ,  $i = 1 : n \in \mathbb{N}^+$  is called an instance of the full-description form  $\hat{f}_\gamma(x)$  at  $\gamma = \mathbf{c}_i$ .  $\square$

*Example 1.* The expression  $ax + b$  is the full-description form of a set of expressions  $\{f_1 = x; f_2 = 1; f_3 = x + 1\}$  with respect to the following set of values for  $a$  and  $b$ :  $\{\{a = 1, b = 0\}, \{a = 0, b = 1\}, \{a = 1, b = 1\}\}$ .

In the original definition (eq. 1), it is implied that changes in dynamic problems can be represented as changes in the parameter space. This has been described more explicitly in [12] and implemented by us in [9]. In this paper this concept will be formulated in a more formal level and will be explicitly made applicable to DTPs: *most common types of changes in dynamic (time-linkage) problems can be represented as changes in the parameter space if we can formulate the problem in a general enough full-description form.* For example, a function-switching change from  $f(x)$  at  $t = 0$  to  $g(x)$  at  $t \geq 1$ ,  $t \in \mathbb{N}^+$  can be expressed as  $\hat{f}(x) = a(t)f(x) + b(t)g(x)$  where  $a(t)$  and  $b(t)$  are two time-dependent parameters given by  $\begin{cases} a(t) = 1 \text{ and } b(t) = 0 \text{ if } t = 0 \\ a(t) = 0 \text{ and } b(t) = 1 \text{ otherwise} \end{cases}$

In real-world problems, changes in the parameters are usually controlled by some specific time-dependent rules or functions<sup>5</sup>. Some time-dependent rules may also have the time-linkage feature, i.e. they also take solutions found by the algorithm up to the current time step as their parameters. The dynamic rules (non-time-linkage and time-linkage) can be defined mathematically as follows.

**Definition 3 (Time-linkage dynamic driver).** *Given a tuple  $\langle t, \gamma_t, X_{\hat{f}}^{G[1,t]} \rangle$  where  $t \in \mathbb{N}^+$  is a change step variable;  $\gamma_t \in \mathbb{R}^m$  is an  $m$ -element vector containing all  $m$  time-dependant parameters of a full-description form  $\hat{f}$  at  $t$ ; and  $X_{\hat{f}}^{G[1,t]}$  is a set of  $k$   $n$ -dimensional solutions achieved by applying an algorithm  $G$  to solve  $\hat{f}$  during the period  $[1, t]$ ; a dynamic rule  $D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t)$  can be seen as a mapping  $\mathbb{R}^m \times \mathbb{R}^{n \times k} \times \mathbb{N}^+ \longrightarrow \mathbb{R}^m$ . The output of  $D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t)$  is a vector  $\gamma_{t+1} \in \mathbb{R}^m$*

$$\gamma_{t+1} = D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t) \quad (3)$$

If all elements of  $\gamma_{t+1}$  are used as values for  $m$  time-dependant parameters of  $\hat{f}$  at the next change step  $t + 1$  then we call  $D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t)$  the time-linkage dynamic driver<sup>6</sup> of the time-dependant parameters set  $\gamma_t$  of  $\hat{f}$ .  $\square$

<sup>5</sup> For example, in some real-world systems changes of parameters can be represented by a linear, chaotic or other non-linear equations of the time variable  $t$ . Dimensional changes can also be represented as changes in the parameter given that the maximum number of variables is taken into account in the full-description form. For example, the function  $\sum_{i=1}^n x_i^2$  with dimension  $n$  varies from 1 to 2 can be represented as the full-description form  $\sum_{i=1}^2 b_i(t)x_i^2$  with  $b_i(t) \in \{0, 1\}$  depending on  $t$ .

<sup>6</sup> *Dynamic driver* is a term used by Morrison[10] in his work to refer to the logistic function in his benchmark. We can borrow this term to refer to any rules/functions that control the dynamic of a dynamic problem.

There are cases where  $X_{\hat{f}}^{G[1,t]}$  does not have any influence on the future of  $\hat{f}$ . In these cases  $D(\gamma_t, X_{\hat{f}}^{G[1,t]}, t)$  becomes a regular dynamic driver with no time-linkage feature.

**Definition 4 (Time unit).** *In a dynamic optimization problem, a time unit, or a unit for measuring time periods in the problem, represents the time durations needed to complete one function evaluation of that problem.<sup>7</sup> The number of evaluations (or time units) that have been evaluated so far in a DOP is measured by the variable  $\tau \in \mathbb{N}^+$ .*

**Definition 5 (Change step and change frequency).** *In a DOP, a change step represents the moment when an environmental change happens. The number of change steps that have happened so far in a dynamic environment is measured by the variable  $t \in \mathbb{N}^+$ . Obviously  $t$  is a time-dependent function of  $\tau$ - the number of evaluations made so far;  $t(\tau) : \mathbb{N}^+ \rightarrow \mathbb{N}^+$ . Because  $t(\tau)$  is also a time-dependent parameter of the DOP  $f$ , its dynamic is controlled by a problem-specific time-based dynamic driver  $D_T(t(\tau), \tau)$ .  $D_T$  decides the frequency of change of the problem and can be described as follows:*

$$\begin{cases} t(1) = 1 \\ t(\tau + 1) = t(\tau) + 1 \text{ if } D_T(t(\tau), \tau) = \text{true} \\ t(\tau + 1) = t(\tau) \quad \text{otherwise} \end{cases} \quad (4)$$

**Definition 6 (Dynamic time-linkage optimization problem).** *Given a tuple  $\langle \hat{f}, \hat{C}, D_P, D_D, D_T, G \rangle$ , a dynamic time-linkage optimization problem in the period  $[1, \tau^{end}]$  function evaluations,  $\tau^{end} \in \mathbb{N}^+$ , can be defined as*

$$\max \left\{ \sum_{\tau=1}^{\tau^{end}} \hat{f}_{\gamma(t_{\tau}, X_{\hat{f}}^{G[1,t]})}(\mathbf{x}_t) \right\} \quad (5)$$

---

<sup>7</sup> This has been mentioned in [1] and [12].

subject to  $\widehat{C}^{i=1:k \in N^+}(\mathbf{x}_t, t_\tau) \leq 0$ ; and  $\mathbf{l}(t_\tau, X_{\widehat{f}}^{G[1,t]}) \leq \mathbf{x}_t \leq \mathbf{u}(t_\tau, X_{\widehat{f}}^{G[1,t]})$

where

$\widehat{f}$  is the full-description form of the objective function  
 $\widehat{C}^1 \dots \widehat{C}^k$  are the full-description forms of  $k$  dynamic constraints<sup>8</sup>  
 $D_P$  is the dynamic driver for parameters in objective and constraint functions (see below)  
 $D_D$  is the dynamic driver for domain constraints (see below)  
 $D_T$  is the dynamic driver for times and frequency of changes (eq. 4)  
 $G$  is the algorithm used to solve the problem  
 $\tau \in [1, \tau^{end}] \cap \mathbb{N}$  is the number of function evaluations done so far  
 $t_\tau$ , or  $t(\tau) \in \mathbb{N}^+$  is the current change step;  $t(\tau)$  is controlled by  $D_T$  (eq. 4)  
 $X_{\widehat{f}}^{G[1,t]}$  is the set of solutions achieved by applying the algorithm  $G$  to solve  $\widehat{f}$  during  $[1, t]$   
 $\gamma_{t_\tau} \in \mathbb{R}^p$  is the time-dependant parameters of  $\widehat{f}$  and  $\widehat{C}^i$ ;  $\gamma_{t_{\tau+1}} = D_P(\gamma_{t_\tau}, X_{\widehat{f}}^{G[1,t]}, t)$   
 $\mathbf{l}(t_\tau), \mathbf{u}(t_\tau) \in \mathbb{R}^n$  are domain constraints;  $\begin{cases} \mathbf{l}(t_{\tau+1}) = D_D(\mathbf{l}(t_\tau), X_{\widehat{f}}^{G[1,t]}, t_\tau) \\ \mathbf{u}(t_{\tau+1}) = D_D(\mathbf{u}(t_\tau), X_{\widehat{f}}^{G[1,t]}, t_\tau) \end{cases}$

The new definition brings us some advantages. Firstly, we can now classify DTPs based on three distinguished components: the full-description forms (determine the statics), the dynamic drivers (determine the dynamics), and the solvers (determine the future problem). Secondly, it supports an important yet not fully considered feature of DTP instances: algorithm-dependency. Finally it represents a general case when solutions from multiple previous steps can affect future performance. The definition can also be used to represent other types of DOPs.

## 4 Time-deceptive and the anticipation approach

According to [5], a dynamic problem is said to be time-deceptive toward an optimizer if the problem is time-linkage and the optimizer cannot efficiently take into account this time-linkage feature during its optimization process.

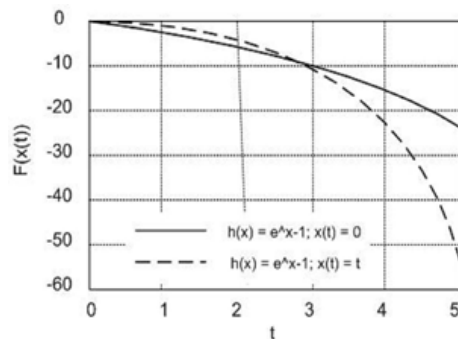
Bosman[3] illustrates this property by proposing the following benchmark:

$$\text{given } n = 1; h(x) = e^x - 1; \max_{x(t)} \left\{ \int_0^{t^{end}} f(x(t), t) dt \right\} \quad (6)$$

$$f(x_t, t) = \begin{cases} -\sum_{i=1}^n (x(t)_i - t)^2 & \text{if } 0 \leq t < 1 \\ -\sum_{i=1}^n \left[ (x(t)_i - t)^2 + h(|x(t-1)_i|) \right] & \text{otherwise} \end{cases}$$

The benchmark problem above is a DTP because for any  $t \geq 1$ , the current value of  $f(x, t)$  depends on  $x(t-1)$  found at the previous time step.

<sup>8</sup> These also include equality constraints because any equality constraint  $c(x) = 0$  can be transformed into an inequality  $|c(x)| - \varepsilon \leq 0$  with a small value  $\varepsilon$ .



**Fig. 1.** This figure (reproduced from [3]) illustrates the time-deception property. We can see that the trajectory of  $f(x_t)$  when we optimize the present (dash line, with optimum solution  $x(t) = t$ ) is actually worse than the trajectory of  $f(x_t)$  with a simple solution  $x(t) = 0$  (the solid line). To solve this problem to optimality, we need to use a predictor to predict the trajectory of function values given different outcomes of current solutions, then choose the one that give us the maximum profit in the future.

An interesting property is revealed when we try to optimize the above problem using the traditional approach: optimizing the present. That property is: *the trajectory formed by optimum solutions at each time step might not be the optimal trajectory*. For example, in figure 1 we can see that the trajectory of  $f(x^*, t)$  when we optimize the present (with optimum solution  $\mathbf{x}^*(t) = t$  at the time step  $t$ ) is actually worse than the trajectory of a  $f(x^0, t)$  with a simple solution  $\mathbf{x}^0 = 0 \forall t$ . It means that the problem is deceptive because an optimizer following the traditional approach is not able to take into account the time-linkage feature.

Bosman [3] [5] suggested that DTPs can be solved to optimality by estimating the values of the function for future times given a trajectory  $\cup_{t=0}^{t_{now}} \{f_t, t\}$  of history data and other previously evaluated solutions. From that estimation, we can choose a future trajectory with optimal future function values. In other words, it is suggested that time-linkage problems can be "solved to optimality" by prediction methods and the result could be "arbitrarily good" if we have a "perfect predictor"[3] [4] [5]<sup>9</sup>. The authors also made some experiments on the benchmark problem mentioned in eq. 6 and on the dynamic pickup problem, showing that under certain circumstances prediction methods do help to improve the performance of the tested algorithms.

<sup>9</sup> A predictor, as defined in [5, line 8-12, pg 139], is "a learning algorithm that approximates either the optimization function directly or several of its parameters... When called upon, the predictor returns either the predicted function value directly or predicted values for parameters". Hence, perfect predictors should be ones that can predict values exactly as the targets.

## 5 Can anticipation approaches solve all DTPs?

Contrary to existing belief, we will show below that there might be cases where the hypothesis above does not hold: if during the predicted time span, the trajectory of the future function values changes its function form, it might not be possible to solve the time-linkage problems to optimality because there is not always the possibility to perfectly predict the future.

Let us consider the situation where predictors help achieving optimal results first. At the current time  $t^{now} \geq 1$ , in order to predict the values of  $f(x(t))$  at a future time  $t^{pred}$ , a predictor needs to take the history data, for example the previous trajectory of function values  $Z^{[0, t^{now}-1]} = \cup_{t=0}^{t^{now}-1} \{f_t, t\}$ , as its input. Given that input, a perfect predictor would be able to approximate correctly the function form of  $Z^{[0, t^{now}-1]}$  and hence would be able to predict precisely the future trajectory  $Z^{[t^{now}, t^{pred}]}$  if it has the same function form as  $Z^{[0, t^{now}-1]}$ .

One example where predictors work is the problem in eq. 6. In that problem, for each trajectory of  $x(t)$  the trajectory of  $f(x(t))$  always remains the same. For example with  $x(t) = t$  the trajectory is always  $1 - e^{t-1}$  or with  $x(t) = 0$  the trajectory is always  $-t^2$  (see figure 1). As a result, that problem is predictable.

Now let us consider a different situation. If at any particular time step  $t^s \in [t^{now}, t^{pred}]$ , the function form of  $Z^{[t^{now}, t^{pred}]}$  changes, the predicted trajectory made at  $t^{now}$  to predict  $f(x(t))$  at  $t^{pred}$  is no longer correct. This is because before  $t^s$  there is no information about how the the function form of  $Z^{[t^{now}, t^{pred}]}$  would change. Without such information, it is impossible to predict the optimal trajectory of function values after the switch, regardless of how good the predictor is. It means that the problem cannot be solved to optimality because it is not possible to perfectly predict the future.

To illustrate this situation, let's consider the following simple problem where the trajectory of function values changes over time (illustrated in figure 2).

$$\widehat{F}(x_t) = a_t f(x_t) + b_t g(x_t) + c_t h(x_t) \quad 0 \leq x_t \leq 1 \quad (7)$$

where  $\widehat{F}(x)$  is the full-description form of a dynamic function;  $f(x_t) = x_t$ ;  $g(x_t) = x_t + (d - 2)$ ;  $h(x_t) = x_t + d$ ;  $a_t, b_t$  and  $c_t$  are the time-dependent parameters of  $\widehat{F}(x_t)$ . Their dynamic drivers are set out as follows:

$$\begin{cases} a_t = 1; b_t = c_t = 0 & \text{if } (t < t^s) \\ a_t = 0; b_t = 1; c_t = 0 & \text{if } (t \geq t^s) \text{ and } \left( \widehat{F}_{t^s-1}(x_{t^s-1}^G) \geq 1 \right) \\ a_t = 0; b_t = 0; c_t = 1 & \text{if } (t \geq t^s) \text{ and } \left( \widehat{F}_{t^s-1}(x_{t^s-1}^G) < 1 \right) \end{cases} \quad (8)$$

where  $t^s > 1$  is a pre-defined time step,  $d \in \mathbb{R}$  is a pre-defined constant, and  $x_{t^s-1}^G$  is a single solution produced at  $t^s - 1$  by an algorithm  $G$ . Eq. 8 means that with  $t < t^s$ , the form of  $\widehat{F}(x_t)$  is always equal to  $f(x_t)$ ; with  $t \geq t^s$ , depending on the solution of  $x_{t^s-1}^G$  the form of  $\widehat{F}(x_t)$  would switch to either  $g(x_t)$  or  $h(x_t)$ .

In the above problem, because at any  $t \geq t^s$  the values of  $a_t, b_t$  and  $c_t$  (and consequently the value of the function  $\widehat{F}$ ) depend on the solution found by  $G$  at  $t^s - 1$ , according to the definition in [5] the problem is considered time-linkage.

This problem has a special property: at any  $t < t^s$  one can only predict the value of  $\widehat{F}$  up to  $t^s - 1$ . Before  $t^s$ , history data does not reveal any clue about the switching rule in eq. 8, hence it is impossible to predict (1) whether the function will switch at  $t^s$ ; (2) which value  $x_{t^s-1}^G$  should get to switch  $\widehat{F}(x_t)$  to  $g(x_t) / h(x_t)$  and (3) which form,  $g$  or  $h$ , would provide better future trajectory.

Even worse, even a predictor that can perfectly learn the current function form of the system might still be deceived to provide worse result than not using any predictor while solving this time-linkage problem! Figure 2 illustrates the situations where the best predictor could provide the worst result while the worst predictor could provide better results after  $t^s$ !

Summarizing, the example problem we proposed in this section illustrates a previously unknown class of DTPs where it is not guaranteed to get optimal results because it is impossible to find a perfect predictor to predict the function values. We call this class *time-linkage problems with unpredictable optimal function trajectories*. The example illustrates a special case where any predictor that relies on past data can be deceived and hence provide the worse results than not using predictor at certain time steps. We call these types of problems the *prediction-deceptive time-linkage problems*.

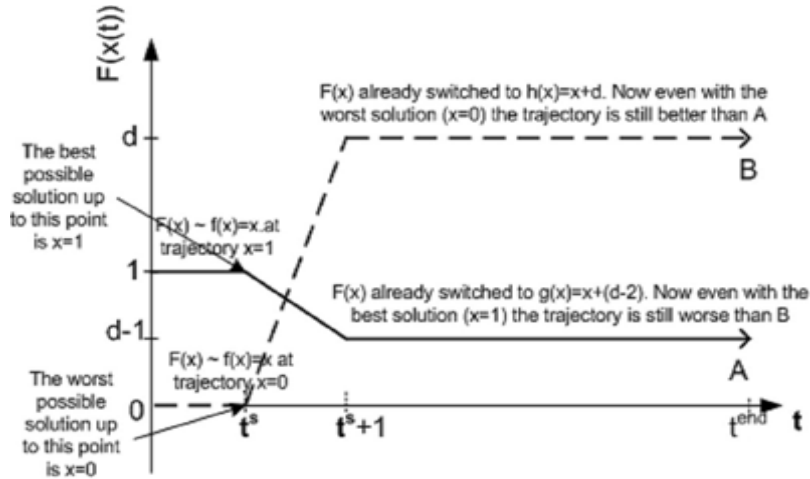
## 6 Another benchmark for time-linkage problems

Currently the only available DTP benchmarks are from [3][5]. Both unfortunately do not cover the prediction-deceptive case. To study prediction-deception and other types of DTPs, besides the problem that we have proposed in eq.7, we also developed a more comprehensive benchmark which covers all three types of dynamics: non-time-linkage, predictable time-linkage and prediction-deceptive time-linkage. Due to limited space we do not introduce details of the benchmark in this paper. Interested readers are referred to the technical report in [11].

## 7 Conclusion

In this paper we have proposed an extended definition for DTPs/DOPs where the problems can be classified based on three components: the full-description forms (determine the static description), the dynamic drivers (determine the dynamics), and the solver (determine how the future problem would be). It is expected that by separating the dynamics/time-linkage feature from the static part of the problem, the definition would make it easier to study the behaviour of DTPs. The definition will also help in generating DTP benchmarks from existing well-studied static benchmarks. It also takes into account aspects as time unit, change step, change frequencies and domain range changes. These details help defining, classifying and characterizing DTPs better.

Secondly, the paper revealed a new class of DTPs that has not been considered before: DTPs with unpredictable optimal function trajectories. We have illustrated that, contrary to previous suggestions, this class of DTPs cannot be solved to optimality because a perfect predictor cannot be found.



**Fig. 2.** This figure illustrates a situation where even the best predictor + the best algorithm (A) still perform worse than the worst predictor + the worst algorithm (B) due to the prediction-deceptive property of the problem in eq.7. Assume that we want to predict the trajectory of  $F(x)$  from  $[0, t^{end}]$ . In case A, the best predictor allows us to predict  $F(x) \sim f(x) = x$  in just only one time step  $[0, 1]$ . With that perfect prediction the algorithm is able to find the best solution  $x = 1$ , which is valid until  $t^s$ . Now at  $t^s$  although the history data tells the predictor that the trajectory must still be  $F(x) \sim f(x) = x$ , according to eq.8 the actual  $F(x)$  does switch to  $g(x) = x + (d - 2)$ , which is the worst trajectory. In other words, the best predictor chose the worst trajectory to follow. On the contrary, in the case B the worst predictor + worst algorithm actually get benefit from the switch: the terrible solution ( $x = 0$ ) they found during  $[0, t^s]$  does help them to switch to  $F(x) \sim h(x) = d + x$ , whose trajectory after  $t^s$  is always better than A regardless of the value of  $x$ .

Finally, we have proposed a time-linkage benchmark problem to study that particular type of time-linkage problems. We also provided a link to a technical report where a more complete set of DTP benchmarks are described.

## Acknowledgement

The authors are grateful to P. Rohlfshagen, T. Ray, L. Xing and three anonymous reviewers for their helpful comments. This work is partially supported by an EPSRC grant (No. EP/E058884/1) on "Evolutionary Algorithms for Dynamic Optimisation Problems: Design, Analysis & Applications", an ORS Award and a School of Computer Science PhD Studentship.

## References

1. Bäck T. On the behavior of evolutionary algorithms in dynamic environments. In *Proc. of the IEEE Intl. Conf. on Evolutionary Computation*, 446- 451, 1998.
2. Bäck T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford Univ. Press, 1996.
3. Bosman P. A. N. Learning, anticipation and time-deception in evolutionary online dynamic optimization. In *Proc. of the GECCO 2005 Conf.*, pp. 39–47, 2005.
4. Bosman P. A. N. & Poutré H. L. Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case. In *GECCO '07: Proc. of the 9th Conf. on Genetic and Evolutionary Computation*, ACM, USA, pp. 1165–1172, 2007.
5. Bosman P. A. N. Learning and Anticipation in Online Dynamic Optimization. In S. Yang, Y.S. Ong and Y. Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, pp. 129-152, Springer-Verlag, Berlin, 2007.
6. Branke J. & Mattfeld D. Anticipation in dynamic optimization: The scheduling case. In M. Schoenauer et al., editors, *Proc. of the Parallel Problem Solving from Nature - PPSN VI*, pp 253-262, Berlin. Springer Verlag, 2000.
7. Dorfman R. An Economic Interpretation of Optimal Control Theory. *American Economic Review*, 59(5), 817-831, 1969.
8. van Hemert J. I. & La Poutre J. A. Dynamic routing problems with fruitful regions: models and evolutionary computation. In X. Yao et al., editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 692-701, Berlin, 2004, Springer Verlag.
9. Li C., Yang S., Nguyen T.T., Yu E.L., Yao X., Jin Y., Beyer H.-G. & Suganthan P.N. Benchmark Generator for CEC 2009 Competition on Dynamic Optimization, Technical report, University of Leicester and University of Birmingham, UK, (2008). URL: <http://www.cs.le.ac.uk/people/syang/ECiDUE/DBG.tar.gz>
10. Morrison R W. *Designing Evolutionary Algorithms for Dynamic Environments*, number ISBN 3-540-21231-0, Springer-Verlag, Berlin, 2004.
11. Nguyen T. T. A benchmark for dynamic time-linkage problems, *Technical Report*, School of Computer Science, University of Birmingham, 2009.
12. Rohlfshagen P. & Yao X. Attributes of combinatorial optimisation. *Proc. of the 7th Int. Conf. on Simulated Evolution and Learning*, pp. 442-451, Springer, 2008.
13. Sistu P. B., Gopinath R. S. & Bequette B.W. Computational issues in nonlinear predictive control, *Comput. Chem. Eng.* 17. pp. 361–367, 1993.