Analysis and Modeling of Control Tasks in Dynamic Systems

Rasmus K. Ursem, Thiemo Krink, Mikkel T. Jensen
Department of Computer Science
Bldg. 540, University of Aarhus
DK-8000 Aarhus C, Denmark
{ursem, krink, mjensen}@daimi.au.dk

Zbigniew Michalewicz
Chief Scientist, NuTech Solutions, Inc.
8401 University Executive Park Dr., Suite 102
Charlotte, NC 28262, USA
zbigniew.michalewicz@nutechsolutions.com

Abstract— **Most applications of evolutionary algorithms (*EAs*) deal with static optimization problems. However, in recent years, there has been a growing interest in time-varying (dynamic) problems, which are typically found in real-world scenarios. One major challenge in this field is the design of realistic test-case generators, which requires a systematic analysis of dynamic optimization tasks. So far, only a few test-case generators (TCGs) have been suggested. Our investigation leads to the conclusion that these TCGs are not capable of generating realistic dynamic benchmark tests. The result of our research is the design of a new TCG capable of producing *realistic* nonstationary landscapes.**

Keywords— **Adaptive control, dynamic problems, real-world problems, test case generator.**[1]

I. Introduction

The ultimate goal in the design of optimization techniques is their application to real-world problems. However, evolutionary algorithms have been applied mainly to static problems even though most real-world problems consist of components that change over time. Evolutionary algorithms have particularly great potential to tackle dynamic problems compared to other iterative search techniques. The primary advantage is that evolutionary algorithms maintain a population of solutions, rather than just a single solution. This provides the potential for a diversity of approaches to problem solving. When the problem changes, to cite a cliché, we don't have all of our "eggs in one basket." If the constraints change and make one solution infeasible, perhaps another reasonable solution in the population will still be feasible. We can examine each solution in the population and determine if any of the currently available alternatives are of value. Further, each solution offers a starting point for discovering new solutions given whatever change has occurred. We don't have to rely on only a single starting point, and we certainly don't have to recompute a new solution starting from *tabula rasa*. If there are any similarities between the old problem and the new problem, it is possible that these will be reflected in the solutions that are present in the population.

Most studies on optimization of dynamic problems fall in one of two categories. They either describe how to handle a specific real-world problem or they introduce novel methods for optimization of dynamic problems. The test problems used in the latter group are often standard problems like the time-varying knapsack problem, a variant of the peak-tracking problem, or the dynamic NK-matching problem.

Recently several authors suggested new test-case generators (*TCGs*) for implementing peak-tracking problems ([1], [2], [3], and [4]). These TCGs are based on deterministic or stochastic updating of peak characteristics such as position, height, and width. Although the introduction of these TCGs was important, no research has been conducted to thoroughly evaluate how well they reflect characteristic dynamics of real-world problems.

Many dynamic problems can be viewed as either observation or control problems. The main difference between these two types of classes is the feedback from the controller to the system (see figure 1).

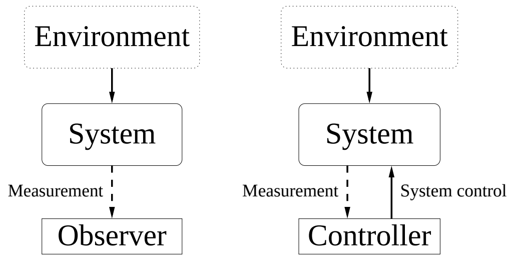

Fig. 1. Left: Observation problem. The environment influences the system; the observer does not affect the system. Right: Control problem. The environment affects the system. The controller and the system interact; decisions made by the controller affects the system.

The objective in observation problems is either to predict and report the values of certain system variables (prediction models) or to process sampled data (signal processing). The difference between these two subclasses is that prediction models use observations from the past to predict the future, whereas signal processing focuses on the extraction of information from recorded data. Typical examples for prediction models are weather forecasting, stock value prediction, and server failure prediction. Signal processing deals with tasks such as speech recognition and noise filtering. Evolutionary algorithms have been applied successfully to several observation problems (e.g. [5], [6], and [7]).

In control problems, a controller has to operate a system and, in many cases, meet a certain output goal. The input

[1]IEEE copyright notice: This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

for this process is provided by sensors that measure the state of the system and its environment. In other words, there is a feedback loop in which the controller changes the system variables that it uses as its own input. If the EA is running while the system is being controlled it actually has the interesting consequence that *the search itself changes the fitness landscape*. Evolutionary algorithms have been used to control several dynamic systems (e.g., [8], [9], [10], [11], [12], and [13]).

The recently proposed TCGs focus on how the landscape changes instead of the underlying dynamics. Whether or not these TCGs can model any real-world problems is still an open question; however, since no control parameters are fed back into any of the TCGs, they certainly do not model control problems.

The focus in this paper is on control problems. Note that an observation problem can be viewed as a control problem without controllable parts. The paper is organized as follows. The next section discusses general characteristics of control problems. Section III contains a description of a model for control problems. Section IV presents the new test-case generator and section V covers its implementation details. Section VI provides an extended example (the greenhouse production model) and section VII summarizes the results from this study. A general discussion of EA-related control strategies is given in section VIII. Finally, section IX concludes the paper.

II. Characteristics of Control Problems

A fundamental understanding of typical dynamics in real-world control problems is essential when designing realistic TCGs. The main motivation for our research was to propose a framework for analyzing general characteristics of real-world problems and to suggest a new TCG capable of modeling realistic dynamic problems. In the process of developing the framework we studied several examples from biology, computer science, engineering, and economics. Based on this study, we suggest to classify *control problems* into three categories:

- *Demand meeting.* The objective in demand meeting is the efficient management of resources while matching a certain level of demand from the environment. The focus is more on meeting the demand than on having an efficient production. For instance, it is better that the production at a powerplant is stable at a sufficient level than having an insufficient production at a lower cost. Another type of demand meeting problems involve the management of buffers while meeting a demand. A typical example is the inventory problem in which a stock has to be managed to provide a demanded resource such as steel for a car production.
- *State stabilization.* State stabilization is a special case of demand meeting without buffering. The main task is to anticipate changes in the state and to act accordingly in advance. Conclusively, a successful stabilization requires that the environment is quite predictable and its overall influence on the system is not too stochastic. The auto-piloting of an aircraft and the well-known pole balancing problem are examples of state stabilization problems.
- *Interacting agents and competition systems.* This class is characterized by coadaptive processes, where the success of an agent depends directly on the actions of another agent and vice versa. The agent control is either direct, such as in robot control, or indirect by modification of the agent's environment. Typical examples are competing companies and coevolutionary systems in biology, such as the epidemic control of diseases by vaccination (direct control) and elimination of transmitting hosts, such as mosquitos (indirect control).

A realistic TCG for control problems must be able to generate changing landscapes that correspond to problem characteristics in at least one of these categories.

III. A Model for Control Problems

The traditional engineering approach to control problems is to view the problem as an interaction between the controller and the system being controlled (see figure 2). The control signals at time t are represented by the vector $\mathbf{u}(t)$, the system state is modelled by the vector $\mathbf{x}(t)$, and the system output vector is $\mathbf{y}(t)$. There are several issues, e.g., analog-digital conversion and sample rate, to consider when dealing with real systems, see [14] for a general introduction to control theory. To fully model a real system its environment often has to be modelled as well (see figure 3).

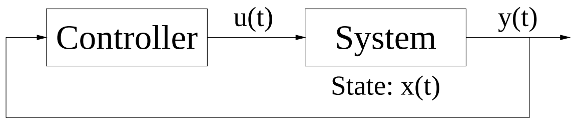

Fig. 2. Controller and the system being controlled. $\mathbf{u}(t)$ is the control signal vector at time t, $\mathbf{x}(t)$ is the system state vector, and $\mathbf{y}(t)$ is the system output.

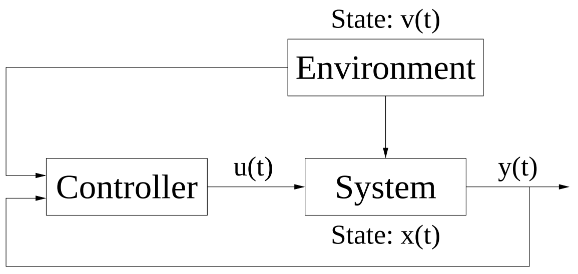

Fig. 3. Controller, the system, and the environment. $\mathbf{u}(t)$ is the control signal vector at time t, $\mathbf{v}(t)$ is the environement state vector, $\mathbf{x}(t)$ is the system state vector, and $\mathbf{y}(t)$ is the system output.

In summary, the model consists of the following four components, which we illustrate by a greenhouse control problem.

First, the *environment* is the immediate surroundings that affect the system. The *environment state* $\mathbf{v}(t)$ represents the variables needed to model the nearby environment and other external components that influence the system.

The greenhouse environment consists of sunlight intensity, outside temperature, and market prices (crop, oil for heating, CO_2).

Second, the *system* is the components that are directly influenced by the controller. Its internal state is modeled by the *system state* vector $\mathbf{x}(t)$. The system state in the greenhouse example consists of the internal temperature, CO_2 level, and the amount of grown crops.

Third, the *performance* of the system is determined by the quality of the system behavior in response to a certain objective, e.g., deviation from a reference value. The performance in the greenhouse is defined as the income from the grown crops minus the expenses used in the production.

Fourth, the *controller* consists of a *decision maker* and a vector of *control signals* $\mathbf{u}(t)$. The decision maker determines the control signals based on information from the environment, the system, and the recorded performance. In the greenhouse example the amount of heating, ventilation, and CO_2 injection are controlled directly. The optimal greenhouse control will maximize the profit by minimizing the production costs and maximizing the production.

The change in system state is usually modelled by a number of difference equations of the form:

$$x_i(t+h) = x_i(t) + \Delta x_i(\mathbf{u}, \mathbf{x}, \mathbf{v}, t, h) \tag{1}$$

where x_i is the ith system variable, $\Delta x_i(\cdot)$ is the update function, t is the time, h is the length of a time-step, and $\mathbf{u}$, $\mathbf{x}$, and $\mathbf{v}$ are the control signals, the system state, and the environment state of previous time-steps (sometimes several steps in the past). Many physical systems can be described by nonlinear differential equations, which can be approximated by difference equations using the Euler or, preferrably, the Runge-Kotta method [15]. In this case the update function $\Delta x_i(\cdot)$ of equation (1) is defined according to the used approximation method.

In general, it is not straightforward to draw the line between the system environment, the system, and the controller. A bottom-up strategy might be the best way to describe a control problem. The first step is to identify all relevant variables related to each part of the problem and the performance. There are two kinds of relevant variables, those with *direct* influence on the system state and those that are relevant for the decision making process in the controller. The next step is to assign each variable to a part of the model (control, system, or environment). This decision should be based on the factors that determine the value of a variable: (i) control state variables can be fully controlled; (ii) system state variables are directly influenced by the control, as well as other factors from the environment or the system itself; (iii) environment variables cannot be controlled, but might influence the system.

IV. The new TCG

The goal in introducing a new TCG was to allow easy implementation of realistic benchmark problems. To achieve this we focused on the underlying mechanisms that generate a time-varying fitness landscape, rather than the "landscape oriented" change of peak characteristics as implemented by other TCGs. Furthermore, we emphasized that the problem analysis should support the implementation of the problem in the new TCG. The secondary goal was to propose a flexible test-case generator, that would support both a set of standard benchmark problems and allow the implementation of realistic real-world control problems.

A TCG-modeled problem is defined by the control, system, and environment variables. Further, the performance is defined as a fitness function of the variables[2]. Central to our new TCG are a number of properties that characterize the dynamics of the variables in the system and the environment state. A property that affects the future value of a variable is called an *effector*. Properties associated with each variable are *domain*, *periodicity*, *stochasticity*, *drift*, and *dependency*; the last four are effectors. Each system and environment variable requires the specification of all these properties, whereas the control variables are only defined by the domain. The properties of the variables are defined as follows:

Domain

The domain of a variable defines its set of possible values; it is characterized by the type (categories, discrete values, continuous values) and the range of values.

Periodicity

This property describes the temporal correlation between successive values of a variable, i.e., whether the value of the variable follows a repeating pattern.

This property can be modelled by a Fourier function (see equation (2)).

$$\begin{aligned} f(t', \mathbf{a}, \mathbf{A}, \mathbf{b}, \mathbf{B}) = &\sum_{i=1}^{|A|} a_i \cos(2\pi A_i t') + \\ &\sum_{i=1}^{|B|} b_i \sin(2\pi B_i t'), \end{aligned} \tag{2}$$

where $\mathbf{a}, \mathbf{A}, \mathbf{b}$, and $\mathbf{B}$ are parameter vectors for the generated periodic function. To model fluctuations in the period length we replaced the normal, linearly increasing time t with an artificial, non-linearly increasing time t'. In each time-step the artificial time t' is increased by a small positive value, which is calculated by a "time advancement function" $g(t)$. The new artificial time is defined as $t'_{new} = t' + g(t)$. The phase of the periodic function (equation (2)) can be shifted by setting the initial value t'_0 of the artificial time t'. Figure 4 illustrates a simple periodic function with and without phase-shift.

A periodic function with fixed period length can be modeled by defining $g(t)$ as a constant function such as $g(t) = 0.5$, $g(t) = 1$, or $g(t) = 3$. A decreasing period length can be modeled with an increasing $g(t)$ function. Figures 5, 6, and 7 illustrate three periodicity effectors with the corresponding time advancement functions.

[2]A multiobjective control problem can be implemented by specifying multiple fitness functions.

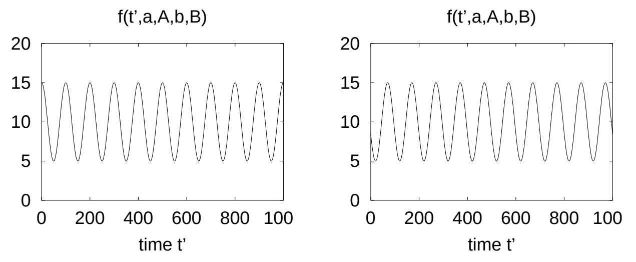

Fig. 4. **a**=[10,5], **A**=[0,0.01], **b**=[], **B**=[], $g(t) = 1$. Left graph, no phaseshift ($t'_0 = 0$). Right graph, phaseshift ($t'_0 = 30$).

Since the TCG variables might require different time advancement function each TCG variable has its own artificial time and time advancement function.

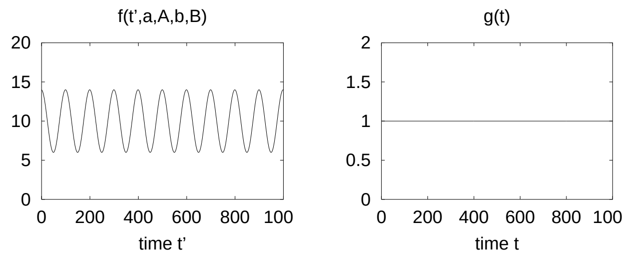

Fig. 5. A periodic function with constant period length (left) $\mathbf{a} = [10, 4], \mathbf{A} = [0, 0.01], \mathbf{b} = [], \mathbf{B} = []$, $t'_0 = 0$, and $g(t) = 1$ (right).

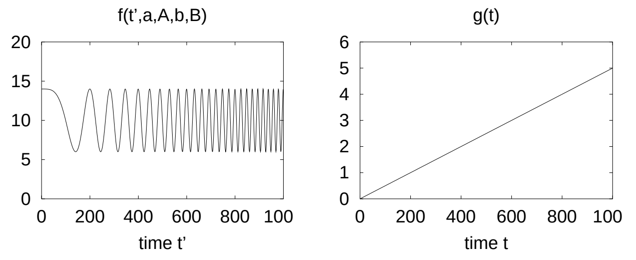

Fig. 6. A periodic function with linearly decreasing period length (left) $\mathbf{a} = [10, 4], \mathbf{A} = [0, 0.01], \mathbf{b} = [], \mathbf{B} = []$, $t'_0 = 0$, and an increasing time-advancement function $g(t) = 0.005t$ (right).

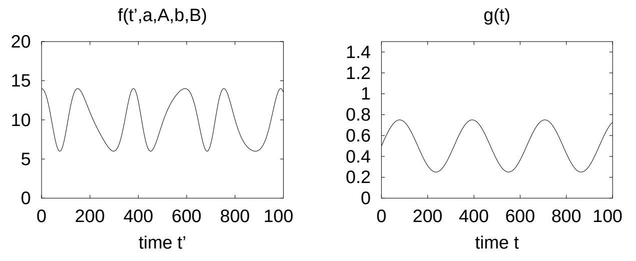

Fig. 7. A periodic function with sinusoidal changing period length (left) $\mathbf{a} = [10, 4], \mathbf{A} = [0, 0.01], \mathbf{b} = [], \mathbf{B} = []$, $t'_0 = 0$, and a periodic time-advancement function $g(t) = 0.5 + 0.25sin(0.02t)$ (right).

A periodicity effector is thus defined by six parameters, the four vectors $\mathbf{a}, \mathbf{A}, \mathbf{b}$, $\mathbf{B}$, which contain the constants for the function, the initial artificial time t'_0, and finally the time advancement function $g(t)$. A combination of a multiple periodic functions can be modeled by specifying more values in the parameter vectors.

Stochasticity

The stochasticity effector models the inherent randomness in a variable. The effector generates random numbers according to a stochastic distribution, e.g., the normal distribution, the uniform distribution, or the binomial distribution. The stochasticity effector is an expression consisting of an arbitrary combination of distribution functions and plain arithmetics. For instance, $N(0, U(0.5, 1.5))$ generates a normal distributed number based on mean 0 and a uniform distributed variance between 0.5 and 1.5.

In some cases a slowly changing stochastic effector (that is an effector in which the value at time $t + h$ is correlated to the value at time t) is needed. To model stochastics, such as random walk, a number of effectors implementing this idea are available in the TCG.

Drift

Drift is present in a variable if the value of the variable has a tendency to change towards one direction only. An illustrative example for drift is the wear out of machinery. A special case of this is *buffer drift*, where a variable represents a stock of some sort, which is gradually emptied. From time to time the stock may be refilled, which temporarily increases its value.

The drift effector consists of a drift function $D(t')$. The drift function can be any expression that defines how the variable is affected by drift. In the drift function, t' measures the time since the beginning of a drift period. Buffered drift is modeled by resetting t' when a drift period is completed. The length of a drift period is determined by an additional function $D_{pl}(t)$, which can be any expression based on plain arithmetics and stochastic functions. Figure 8 illustrates two drift functions.

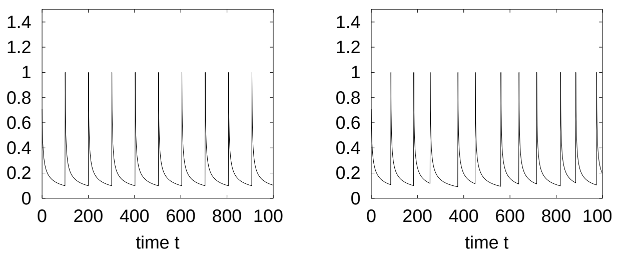

Fig. 8. Two examples of drift functions. Left is $D(t') = 1/\sqrt{t' + 1}$ with constant period length $D_{pl}(t) = 100$ and right $D(t') = 1/\sqrt{t' + 1}$ with variable period length $D_{pl}(t) = N(100, 20)$.

Dependency

The relation between variables is modeled by a network that describes the dependencies in the modeled system. Figure 9 illustrates an example of a network of variables (u_1 to u_l are the control variables, x_1 to x_n denote the system variables, and v_1 to v_m are the environment variables).

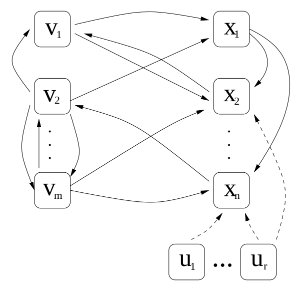

Fig. 9. A problem modeled as a network of variables.

The arcs represent relationships between variables. For instance, the arc from v_1 to x_1 indicates that v_1 affects x_1.

An important aspect of the dependency representation is to distinguish between external and internal effectors. The external effectors represent the influence from external non-modeled events and cover the criteria stochasticity, drift, and periodicity. The internal effectors model the interplay between identified variables and cover the dependency criterion. The interplay between a variable and its neighbors is illustrated in figure 10. The value of each variable is modified by a number of inputs from external and internal effectors. Furthermore, the variable affects other variables by its output effectors.

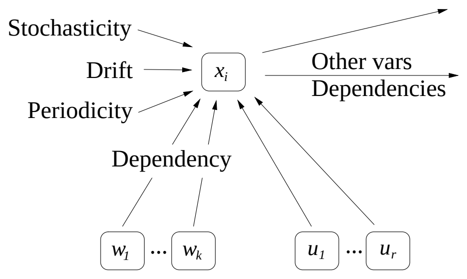

Fig. 10. A variable connected to internal and external effectors. ($w_1 ... w_k$ are the system or environment variables affecting the variable.)

The new value of the variable is calculated by a function of the internal and external input effectors. The variables are updated in parallel, i.e., the values of all variables at time $t + h$ are calculated on the basis of the values at time t.

V. Implementation and use of the TCG

The TCG consists of a simulation shell that contains the current simulation step s, the step length h, the global time t, an array for the performance measures, and three arrays for the control, system, and environment variables. The global time is defined as $t = h \cdot s$. Any TCG variable is modeled by a data structure that contains its current value, a record of past values, the domain of the variable, and the parameters for drift, stochasticity, and periodicity. Moreover, the data structure contains a number of internal variables that are used for management of the TCG variables (calculation of new state, resetting, etc.).

The procedures for updating the TCG variables:

- *Update*, *FinalizeUpdate*

Update calculates the values from the periodicity, stochasticity, and drift functions. These values are used to calculate the new value of the TCG variable, which is stored internally until all variables have been calculated for the next time-step. The TCG then performs the parallel update by calling *FinalizeUpdate* for each of the system and environment variables. *FinalizeUpdate* then copies the new value to the internal variable holding the current value.

- *BackupValue*, *RestoreValue*

These methods are used to backup and restore the value of the TCG variable and all its internal variables. Since the TCG must evaluate the controllers from the same starting state the complete state of the TCG must be stored and restored between each evaluation.

- *SetValue*, *GetValue*, *ResetValue*

Set and get the value of the variable. *ResetValue* sets the variable to the value of the initial state of the TCG ($t = 0$).

The procedures for operating the TCG are:

- *UpdateTCG*

This procedure updates the state of the TCG for a given number of time-steps. The update is based on control values from the controller passed to the procedure. The pseudocode for the procedure is listed in figure 11. First, the TCG advances the simulation step by one and the global time by h. Then it acquires the control values from the given controller, and sets the corresponding control variables. Afterwards, the TCG calculates the new values for the system and environment variables. Finally, the TCG updates the states by calling the *FinalizeUpdate* for each of the TCG variables.

```
UpdateTCG(controller, time-steps)
    for (i=0; i<time-steps; i++) {
        s++
        t := h*s
        controlvalues = controller.GetControl()
        for (i=0; i<|controlvariables|; i++)
            controlvariables[i].setValue(controlvalues[i])
        for each system and environment variable
            variable.Update()
        for each system and environment variable
            variable.FinalizeUpdate()
    }
```

Fig. 11. The pseudocode for the UpdateTCG procedure.

- *BackupTCGState*, *RestoreTCGState*

These procedures backup and restore the state of all variables in the TCG. They are used by *GetFitness* to ensure that the controllers are evaluated from the same starting position.

- *ResetTCG*

Resets the TCG to the initial state ($t = 0$) by calling ResetValue on all variables.

- *GetFitness* and *CalcFitness*

In order to calculate the fitness of a controller the TCG simulates the system for a given number of time-steps. In this process *GetFitness* measures the performance in each time-step, which is combined finally to a single fitness value by the *CalcFitness* procedure. The mapping from performance measurements to the fitness value has to be defined as a part of the input to the TCG, because it varies from system to system. The pseudocode for *GetFitness* is listed in figure 12.

```
GetFitness(controller, time-steps)
    BackupTCGState()
    for (i=0; i<time-steps; i++) {
        performancevalues[i] =
            GetCurrentSystemPerformance()
        UpdateTCG(controller,1)
    }
    RestoreTCGState()
    return CalcFitness(performancevalues)
```

Fig. 12. The pseudocode for the GetFitness procedure.

Figure 13 illustrates an example of how the TCG can be used in connection with an EA. In this example the EA evaluates the evolved controllers for three time-steps. Then the EA selects the best controller and uses it to control the system for one time-step. Finally, the evolutionary operators generate the next generation of controllers.

```
EA main
  TCG.Reset()
  initialize population of controllers
  while (not(termination condition)) {
    for each controller in the population{
      controller.fitness =
        TCG.GetFitness(controller, 3)
    }
    UpdateTCG(best controller, 1)
    Apply EA operators to the population
  }
```

Fig. 13. An example of an EA using the TCG.

Figures 14 illustrates the exploration that an EA carries out through the first three time-steps. In each time-step the controllers are evaluated from the same starting state, which is the state determined by the best controller of the previous iteration.

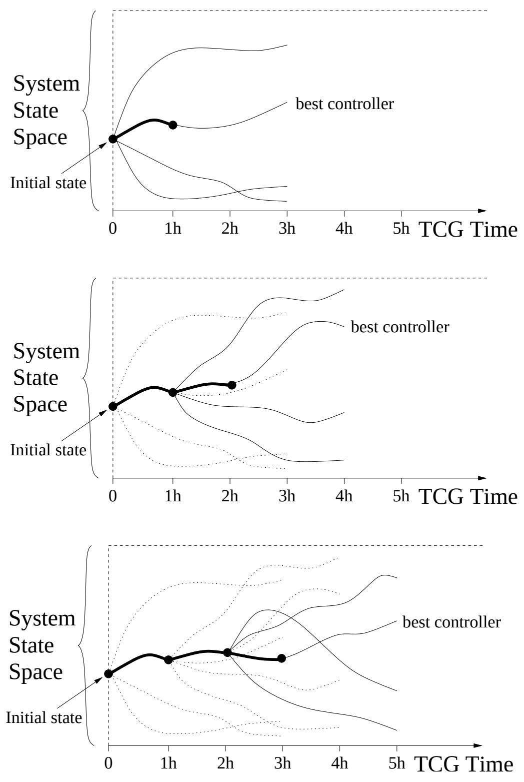

Fig. 14. Example of state space exploration at TCG-time $t = 0$, $t = h$, and $t = 2h$. Thin lines represent controller exploration of the current time-step, thin dotted lines are previous explored control strategies, and thick lines are actual control as it was performed by the selected controller.

VI. The Greenhouse Production Model

This section demonstrates the potential of the new TCG by an implementation of a greenhouse model. Although the model is a simplification, it is sufficient to demonstrate the capability of the new TCG, and produce dynamic landscapes which cannot be achieved with existing TCGs for dynamic problems.

The greenhouse is modeled as follows:

(i) Control variables
- heating u_{heat}
- ventilation u_{vent}
- addition of carbon-dioxide u_{CO_2}

(ii) System variables
- temperature inside the greenhouse x_{temp}
- carbon-dioxide level inside the greenhouse x_{CO_2}
- amount of grown crop x_{crop}

(iii) Environment variables
- temperature of the greenhouse environment v_{temp}
- sunlight intensity v_{sun}
- market prices of:
  - the crop v_{pcrop}
  - the oil for the heating v_{pheat}
  - the CO_2 gas v_{pCO_2}

(iv) Performance
- profit p

In the model each day corresponds to 100 TCG-time-steps and one "year" corresponds to 10 days. This short year was chosen to have a quick variation from summer to winter, meaning shorter simulation runs. The short year in the model is acceptable because the day-to-day correlation is not important.

Domain

All variables are real-valued. Their domains are specified in table I.

Periodicity

The variables v_{temp}, v_{sun}, and v_{pcrop} are periodic. v_{temp} and v_{sun} reflects the daily and seasonal changes of sunlight and temperature, while v_{pcrop} follows the seasonal change in vegetable prices. This was modeled by a periodic effector of two overlaid cosine functions with different angular velocities. The parameters are listed in table I. For v_{temp} the values are: $\mathbf{A} = [0.01, 0.001]$, $\mathbf{a} = [7, 9]$, $t'_0 = -10$, and $g(t) = 1$, which corresponds the periodic function in equation (3). The phase for v_{temp} is shifted slightly compared to v_{sun}, which is to model the time it takes the sun to heat the environment. This is achieved by letting the artificial time t' of v_{temp} start at the value $t'_0 = -10$, and advancing it by 1 in every time-step of the TCG.

$$v_{temp,period}(t') = 7\cos(2\pi \cdot 0.01t') + 9\cos(2\pi \cdot 0.001t') \tag{3}$$

Stochasticity

All the environment variables are influenced by some degree of stochasticity. However, values only change marginally between two time-steps. For instance, the outside temperature v_{temp} does not change much from minute to minute.

The stochastic component of the environment variables is modeled by adding a small random value to the value from the previous time-step. The stochasticity effector for v_{temp} is displayed in equation (4).

$$v_{temp,stoch}(t) = \min(\max(v_{temp,stoch}(t-1) + U(-0.5, 0.5), -4.0), 4.0), \tag{4}$$

where $U(-0.5, 0.5)$ generates a uniformly distributed number between -0.5 and 0.5. Table I contains the functions for the stochasticity effectors. The max and min functions ensure that the stochastic values stay in fixed intervals. The first stochastic values are calculated by setting $v_{temp,stoch}(0) = 0$, $v_{sun,stoch}(0) = 0$, $v_{pcrop,stoch}(0) = 0$, $v_{pheat,stoch}(0) = 0$, and $v_{pCO_2,stoch}(0) = 0$.

Drift

Drift is not present in any of the variables.

Dependency

The relationship between the variables is illustrated in figure 15.

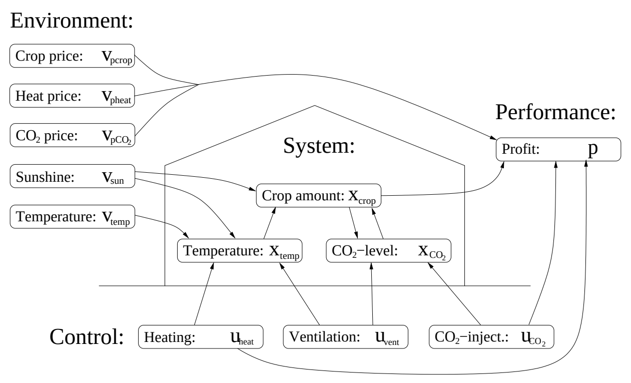

Fig. 15. The dependencies in the greenhouse model.

The environment variables are updated according to equation (5) to (9).

TABLE I

Values and functions for the domain, periodicity, and stochasticity. A '*' in the Init column indicates that the initial value is calculated from the update rule of the variable.

	Domain			Periodicity				Stochasticity	
Variable	Init	Min	Max	**a**	**A**	t'_0	$g(t)$	Init	stochastic function
u_{heat}	0	0	5						
u_{vent}	0	0	1						
u_{CO_2}	1	0	4						
x_{temp}	18	-20	50						
x_{CO_2}	4	0	10						
x_{crop}	1	0	∞						
v_{temp}	*	-20	40	[7, 9]	[0.01, 0.001]	-10	1	0	$\min(\max(v_{temp,stoch}(t-1) + U(-0.5, 0.5), -4.0), 4.0)$
v_{sun}	*	0	8	[4, 2]	[0.01, 0.001]	0	1	0	$\min(\max(v_{sun,stoch}(t-1) + U(-0.25, 0.25), -1.0), 1.0)$
v_{pcrop}	*	0	30	[0, -3]	[0.01, 0.001]	0	1	0	$\min(\max(v_{pcrop,stoch}(t-1) + U(-0.01, 0.01), -5.0), 5.0)$
v_{pheat}	*	0	3					0	$\min(\max(v_{pheat,stoch}(t-1) + U(-0.001, 0.001), -0.5), 0.5)$
v_{pCO_2}	*	0	3					0	$\min(\max(v_{pCO_2,stoch}(t-1) + U(-0.001, 0.001), -0.5), 0.5)$

$$\begin{aligned} v_{temp}(t) &= 10.0 + v_{temp,period}(t) + \\ &\quad v_{temp,stoch}(t) \quad (5) \\ v_{sun}(t) &= 1.0 + v_{sun,period}(t) + \\ &\quad v_{sun,stoch}(t) \quad (6) \\ v_{pcrop}(t) &= 22.0 + v_{pcrop,period}(t) + \\ &\quad v_{pcrop,stoch}(t) \quad (7) \\ v_{pheat}(t) &= 2.5 + v_{pheat,stoch}(t) \quad (8) \\ v_{pCO_2}(t) &= 2.5 + v_{pCO_2,stoch}(t) \quad (9) \end{aligned}$$

The functions $v_{temp,period}(t)$ and $v_{temp,stoch}(t)$ refers to the periodic and stochastic values for the calculation of v_{temp} at time-step t (see table I).

Figure 16 illustrates the values of v_{temp}, v_{sun}, and v_{pcrop}. Each time-step represents a period of 10 days. The crop price shows a clear seasonal variation, while sunshine and temperatures show a daily and seasonal variation with small stochastic variation.

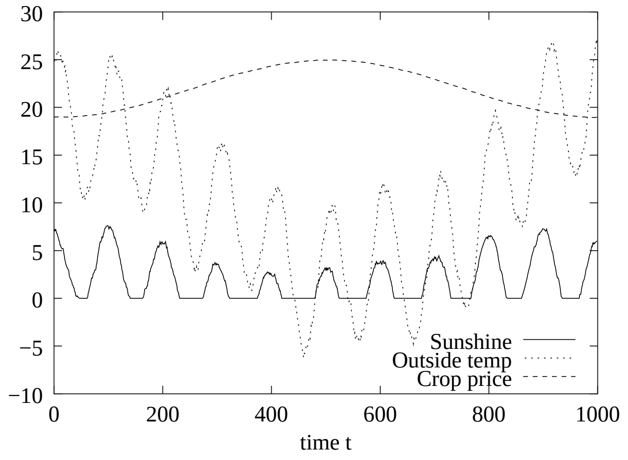

Fig. 16. Three of the environment variables of the greenhouse system.

The system variables depend on each other, on the environment variables, and on the control variables. They are updated incrementally using difference equations of the form:

$$x(t+h) = x(t) + \Delta x(\mathbf{u}, \mathbf{x}, \mathbf{v}, t, h)$$

where x is either x_{crop}, x_{temp}, or x_{CO_2}.

Equation (10) displays the Δ-function for the temperature in the greenhouse (x_{temp}). x_{temp} is controlled by the heating u_{heat} and the ventilation u_{vent}. The ventilation can be used to control the heat exchange with the environment. The minimal heat exchange rate is the constant k_1, which models the insulation value of the glass in the greenhouse. The greenhouse is also heated by the sun v_{sun}. The temperature increase caused by the sun is scaled by the constant k_2, which models how dependent the greenhouse temperature is on the sunlight intensity. The constants for all dependency functions are listed in table II.

$$\begin{aligned} \Delta x_{temp} = u_{heat} + (k_1 + u_{vent})(v_{temp} - x_{temp}) + \\ k_2 v_{sun} \end{aligned} \quad (10)$$

The change of CO_2 is modeled by the formula in equation (11). The CO_2 level decreases when the plants grow. It can be increased by injecting CO_2 (u_{CO_2}) or by ventilation (u_{vent}), whenever the inside CO_2 level is lower than the environmental level. In the latter case this also affects the indoor temperature (x_{temp}). k_3 models the rate of CO_2 consumption by the plants. k_4 is the environmental CO_2 level.

$$\begin{aligned} \Delta x_{CO_2} = k_3 \max(\Delta x_{crop}, 0) + u_{CO_2} + \\ u_{vent}(k_4 - x_{CO_2}) \end{aligned} \quad (11)$$

The growth of the plants is limited by the amount of available resources. Necessary resources are carbon-dioxide (x_{CO_2}), sunlight (v_{sun}), and temperature (x_{temp}), which may not be too cold or hot. The first three lines of equation (12) yield a positive value if the necessary resources are available and the temperature allows the crops to grow. The fourth line is negative if the temperature is either too high or too low. The interpretation of the constants in equation (12) is as follows: k_5 is the maximal growth allowed by the temperature, k_6 is the optimal temperature for growth, k_7 is the maximal amount of CO_2 that can be consumed by the plants, k_8 is the maximal sunlight intensity that can be used by the plants, and k_9 is the rate of decrease in plant-biomass when the temperature is too extreme.

$$\begin{aligned} \Delta x_{crop} = \min(&\max(k_5 - |x_{temp} - k_6|, 0), \\ &\min(x_{CO_2}, k_7), \\ &\min(v_{sun}, k_8)) - \\ &k_9 \min(k_5 - |x_{temp} - k_6|, 0) \end{aligned} \quad (12)$$

The profit in a time-step is defined by:

$$p = v_{pcrop}\Delta x_{crop} - v_{pheat}u_{heat} - v_{pCO_2}u_{CO_2}$$

TABLE II
Constants for the dependency functions.

$k_1 = 0.1$	$k_2 = 0.2$	$k_3 = 1$
$k_4 = 4$	$k_5 = 8$	$k_6 = 26$
$k_7 = 8$	$k_8 = 7$	$k_9 = 0.1$

The initial values of the variables at time-step $t = 0$ are shown in table I. The initial values of the environment variables need not to be specified explicitly, because they are calculated directly from the periodicity and stochasticity of the variable.

In the investigations of the greenhouse example we used a standard GA to search the space of possible control-settings. The TCG was connected with the standard GA as described in the previous section (see figure 13). For each individual, a control setting was simulated for six time-steps. The best of these settings was then used to control the greenhouse for one time-step.

Figures 17 and 18 show the control and system variables over 200 time-steps. At night (TCG time-step 150) the heating is turned on to avoid freezing damage to the crop. At dawn (TCG time-step 170) the CO_2 supply is turned on to support plant growth, while the heating is kept on to rapidly increase the temperature so the optimal growth conditions are reached as fast as possible. When the temperature is high enough (time-step 185) the heating is turned off and ventilation is turned on to keep the temperature down, as the greenhouse is heated by the sun. Late afternoon (time-step 220), when the sunshine intensity decreases, the CO_2 supply is turned off and ventilation is increased. This is a cheaper way of supplying CO_2 when the temperature outside is not too low and a moderate quantity of CO_2 is needed. After sunset (time-step 240) the temperature in the greenhouse decreases towards the damaging level of the crops. The heating is then turned on and the cycle is repeated.

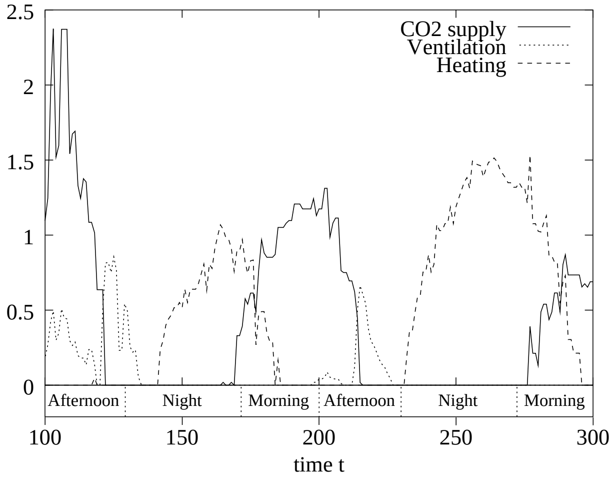

Fig. 17. The controls found by the GA over a period of 200 time-steps (representing 2 days). The controls follow a pattern of cool nights with the heating turned on and warm sunny days with ventilation and CO_2 supply.

Sample plots of the optimization landscape are shown in figure 19 and 20. The figures show the expected profit for the next six time-steps (not the long term profit). Since the greenhouse is a 3-dimensional control problem (u_{heat}, u_{CO_2}, and u_{vent}) one of the variables had to be fixed to produce the plots. The fixed variable was set to the best value suggested by the GA.

Figure 19 illustrates the profit as a function of CO_2 injection (u_{CO_2}) and amount of heating (u_{heat}); the amount of ventilation (u_{vent}) was fixed.

In connection with the design of the greenhouse example we developed a simple rule-based controller that implemented three simple rules; (i) turn up the heat if it is too cold inside, (ii) ventilate if it is too hot, and (iii) inject CO_2 if the level inside is too low. This simple controller

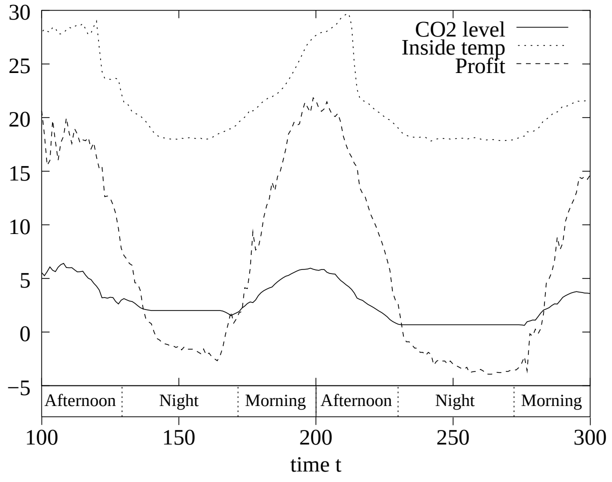

Fig. 18. Some of the system variables and the performance (profit) over a period of 200 time-steps. The greenhouse temperature is much warmer at day than at night. The CO_2-level follows this pattern, since it is raised by the controller to allow maximal growth. There is a profit in the daytime, while at night money is lost by paying for heating.

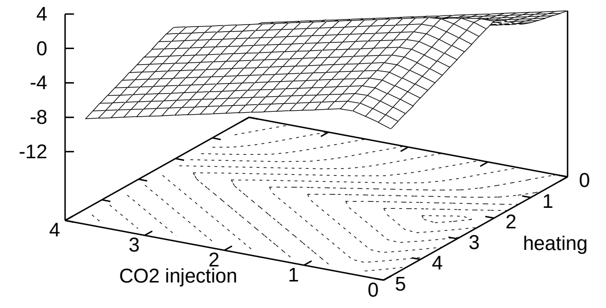

Fig. 19. Expected profit in the next six timesteps. The ventilation was fixed at the value determined by the controller.

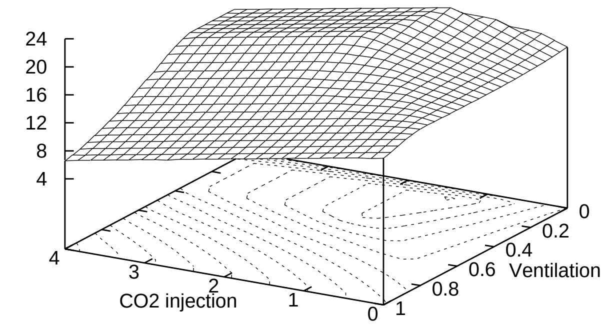

Fig. 20. A fitness landscape from the greenhouse example. The heating was fixed at the value determined by the controller.

performed poorly in the sense that the profit was low and the crops had a tendency to die. Interestingly, the corresponding fitness landscapes were mainly plane-like surfaces tilted towards one of the corners. Hence, the best possible control was to set heating, ventilation, and CO_2 injection to either zero or maximum. These results indicate that poor control leads to simple fitness landscapes, where the optimal control strategy is a corner point in the search space spanned by the control variables.

VII. Discussion of the Greenhouse Model

Our studies of real-world problems was partly motivated by the desire to get a general understanding of the shape and dynamics of their fitness landscapes. Observations from the greenhouse example provided a number of preliminary results. The shape of the landscape is obviously related to the number of optimal control settings at a given time-step. However, even though two good alternative settings are available they might not result in a fitness landscape with two peaks. An example from the greenhouse is the following. Assume that the indoor and outdoor temperature are both near the optimal temperature for crop growth. The crops consume CO_2 when they grow, which lowers the internal CO_2 level. To compensate for the consumed CO_2 the controller can either inject expensive CO_2 or increase the ventilation, which will provide free CO_2 at a lower pace. This seems like two alternative strategies; however, because the CO_2 injection and amount of ventilation are continuous variables, an infinite number of intermediate strategies exists. In the current implementation of the greenhouse, these mixed strategies correspond to a ridge in the fitness landscape with the two extreme strategies at each end of the ridge. Two local optima might appear if a nonlinear relationship between the CO_2 price and the amount of injected CO_2 is present. In this case, it might be optimal to use either a large amount of CO_2 or to avoid a CO_2 injection completely. Nonlinear relations are common in real-world problems. An example is a bulk discount agreement for the unit price of a resource. Another feature that will result in a multimodal control problem is discrete decision making where intermediate solutions are infeasible. For instance, if a robot has to pass an obstacle the controller can either decide to go left or right.

VIII. Discussion of Evaluation and Control Strategies

The evaluation of individuals in real-world problems introduces some technical difficulties. It could be dangerous, expensive, or too time consuming to evaluate all individual in the real system. For instance, it is clearly not an option to let a low-fit individual control a nuclear power plant. Instead, a sufficiently accurate model[3] has to be used for the evaluation process.

The use of models to simulate real-world systems introduces several design issues that play an important role in the choice of algorithm, population size, representation, etc. The most important issue is the maximal allowed response time, which defines how fast the controller must react to ensure proper and safe system control. For instance, driving a car requires rapid responses, which need not be as important in other problems such as the greenhouse. The main problem is that the calculation time for the response might be so long that the system state has changed substantially, thereby making the difference between the model and the real system too big.

[3]EAs offer an interesting way to improve a system model by online modifications, e.g., see [16].

There are several ways to use an EA for control problems. The simplest possible is to evolve the control signals directly. This approach is not used widely, mainly because it requires a quite long response time, because of the time-consuming evolution of the control signals. However, the strategy has been used to control a multiple-burner boiler system [11], a sugar beet press [12], and a greenhouse [10]. In more advanced applications the EA acts as the tuning algorithm for another control strategy. There are several techniques such as fuzzy control, neural net control, genetic programming control, and rule-based control (see [8], [17]). Evolutionary algorithms have also been used to tune traditional engineering controllers such as the well-known PID controller (e.g., [18]).

Another interesting aspect of EA-related control strategies is the possibility of evolving controllers *while* the system is being controlled. If a better controller is evolved it takes over the control of the real system. This technique allows the controller to adapt better to the system and thereby compensate for long-term effects such as wear out of machinery.

IX. Conclusions and Future Work

In this paper we investigated the internal structure and mechanisms of dynamic real-world problems. The main motivation was the need for realistic test problems for optimization of dynamic systems, which are essential for proper evaluation and comparison of EAs. In this context, we suggested a novel TCG for control problems that model the system, its controller, and its environment. We demonstrated the potential of our new test-case generator in a simple modeling example of a crop-producing greenhouse. The resulting fitness landscapes looked surprisingly different from landscapes that can be generated with traditional test-case generators. The landscapes had ridge-like, asymmetric peaks with concave or convex faces, plateaus, and sharp edges (see figures 19 and 20). When the control process was far off from the optimum, the fitness landscapes turned into simple tilted planes. In close vicinity of the optimum, the shape of the landscape changed into more complicated structures.

It seems that the TCGs introduced in [1], [2], [3], and [4] are of little value for modeling realistic dynamic problems. This conclusion is based on the four following observations.

First, the recently introduced TCGs do not model the interactions between the system components. Instead, the TCGs create artificial dynamic problems where the shape and dynamics of the fitness landscape are introduced without any justifying relation to any real problem.

Second, even if the old TCGs could approximate the underlying dynamics by imitating the corresponding landscape one has to analyze the landscape of the real system to imitate it properly. To get an idea of the shape of the fitness landscape a model often has to be developed and implemented, which will make the later imitation of the landscape rather pointless.

Third, the current technical capabilities of the previous TCGs are too limited to produce even simple landscapes like the ones found in the greenhouse example.

Fourth, the previously introduced TCGs do not allow the optimization algorithm to affect the shape of the fitness landscape. This has the consequence that control problems *cannot* be modeled.

These limitations are not present in our TCG, mainly because the landscapes is a *result* of a dynamic system model that mimics the behavior of a real system.

In our future work, we plan to concentrate on a few issues; these include: (i) development of several test problems for each of the three general classes mentioned in the introduction[4], (ii) investigation of discrete dynamic problems such as scheduling and permutation-based problems, and (iii) revision and extension of the modeling framework and the TCG.

Acknowledgements

The authors would like to express their sincere thanks to Bogdan Filipič (Dept. of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia) and Pierré Vadstrup (Grundfos A/S, Denmark) for many valuable comments to earlier versions of this manuscript.

References

[1] Jürgen Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the Congress of Evolutionary Computation*, Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, Eds., Mayflower Hotel, Washington D.C., USA, 6-9 July 1999, vol. 3, pp. 1875–1882, IEEE Press.

[2] Ronald W. Morrison and Kenneth A. De Jong, "A test problem generator for non-stationary environments," in *Proceedings of the Congress of Evolutionary Computation*, Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, Eds., Mayflower Hotel, Washington D.C., USA, 6-9 July 1999, vol. 3, pp. 2047–2053, IEEE Press.

[3] Krzysztof Trojanowski and Zbigniew Michalewicz, "Searching for optima in non-stationary environments," in *Proceedings of the Congress of Evolutionary Computation*, Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, Eds., Mayflower Hotel, Washington D.C., USA, 6-9 July 1999, vol. 3, pp. 1843–1850, IEEE Press.

[4] John J. Grefenstette, "Evolvability in dynamic fitness landscapes: A genetic algorithm approach," in *Proceedings of the Congress of Evolutionary Computation*, Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, Eds., Mayflower Hotel, Washington D.C., USA, 6-9 July 1999, vol. 3, pp. 2031–2038, IEEE Press.

[5] S.C. Ng, C.Y. Chung, S.H. Leung, and A. Luk, "A variable step size algorithm using evolution strategies for adaptive filtering," in *Proceedings of the Congress of Evolutionary Computation*, Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, Eds., Mayflower Hotel, Washington D.C., USA, 6-9 July 1999, vol. 1, pp. 542–545, IEEE Press.

[6] Harald H. Soleng, "Oil reservoir production forecasting with uncertainty estimation using genetic algorithms," in *Proceedings of the Congress of Evolutionary Computation*, Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, Eds., Mayflower Hotel, Washington D.C., USA, 6-9 July 1999, vol. 2, pp. 1217–1223, IEEE Press.

[7] Shu-Heng Chen and Woh-Chiang Lee, "Option pricing with genetic algorithms," in *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, Thomas Bäck, Ed., San Francisco, CA, 1997, Morgan Kaufmann.

[8] Michael Lee and Hideyuki Takagi, "Dynamic control of genetic algorithms using fuzzy logic techniques," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, Stephanie Forrest, Ed., San Mateo, CA, 1993, pp. 76–83, Morgan Kaufman.

[9] Sofiane Oussedik, Daniel Delahaye, and Marc Schoenauer, "Dynamic air traffic planning by genetic algorithms," in *Proceedings of the Congress of Evolutionary Computation*, Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, Eds., Mayflower Hotel, Washington D.C., USA, 6-9 July 1999, vol. 2, pp. 1110–1117, IEEE Press.

[10] Hartmut Pohlheim and Adolf Heissner, "Optimal control of greenhouse climate using real-world weather data and evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, Eds., Orlando, Florida, USA, 13-17 July 1999, vol. 2, pp. 1672–1677, Morgan Kaufmann.

[11] F. Vavak, K. Jukes, and T. C. Fogarty, "Adaptive combustion balancing in multiple burner boiling using a genetic algorithm with variable range of local search," in *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, Thomas Bäck, Ed., San Francisco, CA, 1997, Morgan Kaufmann.

[12] Terence C. Fogarty, Frank Vavak, and Philip Cheng, "Use of the genetic algorithm for load balancing of sugar beet presses," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, Larry Eshelman, Ed., San Francisco, CA, 1995, pp. 617–624, Morgan Kaufmann.

[13] Bogdan Filipič, Tanja Urbančič, and Viljem Križman, "A combined machine learning and genetic algorithm approach to controller design," *Engineering Applications of Artificial Intelligence*, vol. 12, no. 4, pp. 401–409, 1999.

[14] Benjamin C. Kuo, *Automatic Control Systems*, Prentice Hall, 7th edition, 1995.

[15] Robert Adams, *Calculus – a complete course*, Addison-Wesley publishers, 3rd edition, 1995.

[16] F. Vavak, T. C. Fogarty, and P. Cheng, "Load balancing application of the genetic algorithm in a nonstationary environment," *Lecture Notes in Computer Science*, vol. 993, pp. 224–233, 1995.

[17] T. Urbančič, D. Juričić, B. Filipič, and I. Bratko, "Automated synthesis of control for nonlinear dynamic systems," in *IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in Real-Time Control*, Delft, Netherlands, 1992, pp. 605–610.

[18] Chia-Ju Wu, "Genetic tuning of pid controllers using a nural network model: A seesaw example," *Journal of Intelligent and Robotic Systems*, , no. 25, pp. 43–59, 1999.

[4]The problems will be available at our website `www.evalife.dk`.