Test-Case Generator for Nonlinear Continuous Parameter Optimization Techniques

Zbigniew Michalewicz, *Senior Member, IEEE*, Kalyanmoy Deb, Martin Schmidt, and Thomas Stidsen

***Abstract*—The experimental results reported in many papers suggest that making an appropriate *a priori* choice of an evolutionary method for a nonlinear parameter optimization problem remains an open question. It seems that the most promising approach at this stage of research is experimental, involving the design of a *scalable* test suite of constrained optimization problems, in which many features could be tuned easily. It would then be possible to evaluate the merits and drawbacks of the available methods, as well as to test new methods efficiently. In this paper, we propose such a test-case generator for constrained parameter optimization techniques. This generator is capable of creating various test problems with different characteristics including: 1) problems with different relative sizes of the feasible region in the search space; 2) problems with different numbers and types of constraints; 3) problems with convex or nonconvex evaluation functions, possibly with multiple optima; and 4) problems with highly nonconvex constraints consisting of (possibly) disjoint regions. Such a test-case generator is very useful for analyzing and comparing different constraint-handling techniques.**

***Index Terms*—Constrained optimization, evolutionary computation, nonlinear programming, test-case generator.**

I. Introduction

THE general nonlinear programming (NLP) problem is to find $\vec{x}$ so as to

$$\text{optimize } f(\vec{x}), \qquad \vec{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n \tag{1}$$

where $\vec{x} \in \mathcal{F} \subseteq \mathcal{S}$. The *evaluation function* f is defined on the *search space* $\mathcal{S} \subseteq \mathbb{R}^n$, and the set $\mathcal{F} \subseteq \mathcal{S}$ defines the *feasible region*. Usually, the search space $\mathcal{S}$ is defined as an n-dimensional rectangle in $\mathbb{R}^n$ (domains of variables defined by their lower and upper bounds):

$$l_i \leq x_i \leq u_i, \qquad 1 \leq i \leq n$$

whereas the feasible region $\mathcal{F} \subseteq \mathcal{S}$ is defined by a set of p additional constraints $(p \geq 0)$:

$$g_j(\vec{x}) \leq 0, \qquad \text{for } j = 1, \cdots, q$$

and

$$h_j(\vec{x}) = 0, \qquad \text{for } j = q+1, \cdots, p.$$

At any point $\vec{x} \in \mathcal{F}$, the constraints g_j that satisfy $g_j(\vec{x}) = 0$ are called the *active* constraints at $\vec{x}$.

The NLP problem, in general, is intractable: it is impossible to develop a deterministic method for the NLP in the global optimization category which would be better than the exhaustive search [15]. This makes room for evolutionary algorithms, extended by some constraint-handling methods. Indeed, during the last few years, several evolutionary algorithms which aim at complex objective functions (e.g., nondifferentiable or discontinuous) have been proposed for the NLP; a recent survey paper [42] provides an overview of these algorithms.

It is not clear what characteristics of a constrained problem make it difficult for an evolutionary technique (and for any other optimization technique). Any problem can be characterized by various parameters; these may include the number of linear constraints, the number of nonlinear constraints, the number of equality constraints, the number of active constraints, the ratio $\rho = |\mathcal{F}|/|\mathcal{S}|$ of sizes of feasible search space to the whole, and the type of evaluation function (number of variables, number of local optima, existence of derivatives, etc.). In [42], 11 test cases for constrained numerical optimization problems were proposed $(G1$–$G11)$. These test cases include evaluation functions of various types (linear, quadratic, cubic, polynomial, nonlinear) with various numbers of variables and different types (linear inequalities, nonlinear equations, and inequalities) and numbers of constraints. The ratio ρ between the size of the feasible search space $\mathcal{F}$ and the size of the whole search space $\mathcal{S}$ for these test cases vary from 0 to almost 100%; the topologies of feasible search spaces are also quite different. These test cases are summarized in Table I. For each test case, the number n of variables, the type of function f, the relative size of the feasible region in the search space given by the ratio ρ, the number of constraints of each category (linear inequalities *LI*, nonlinear equations *NE*, and inequalities *NI*), and the number a of active constraints at the optimum (including equality constraints) are listed.

The results of many tests did not provide meaningful conclusions, as no single parameter (number of linear, nonlinear, active constraints, the ratio ρ, the type of function, number of variables) proved to be significant as a major measure of difficulty of the problem. For example, many tested methods approached

Manuscript received March 16, 1999; revised November 8, 1999. This work was supported in part by the ESPRIT Project 20288 Cooperation Research in Information Technology (CRIT-2), "Evolutionary Real-Time Optimization System for Ecological Power Control." The work of K. Deb was supported by the Alexander von Humboldt Foundation, Germany.

Z. Michalewicz is with the Department of Computer Science, University of North Carolina, Charlotte, NC 28223 USA, the Institute of Computer Science, Polish Academy of Sciences, 01-237 Warsaw, Poland (e-mail: zbyszek@uncc.edu).

K. Deb is with the Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, Pin 208 016, India (e-mail: deb@iitk.ac.in).

M. Schmidt was with the Department of Computer Science, Aarhus University, DK-8000 Aarhus C, Denmark, and is now with NuTech Solutions Inc., Charlotte, NC 28262 USA (e-mail: martin.schmidt@nutechsolutions.com).

T. Stidsen was with the Department of Computer Science, Aarhus University, DK-8000 Aarhus C, Denmark, and is now with the Institute for Mathematical Modeling, Technical University of Denmark, Lyngby, Denmark (thomas@stidsen.dk).

Publisher Item Identifier S 1089-778X(00)04469-6.

1089–778X/00$10.00 © 2000 IEEE

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

TABLE I
SUMMARY OF 11 TEST CASES. THE RATIO $\rho = |\mathcal{F}|/|\mathcal{S}|$ WAS DETERMINED EXPERIMENTALLY BY GENERATING 1 000 000 RANDOM POINTS FROM $\mathcal{S}$ AND CHECKING WHETHER THEY BELONG TO $\mathcal{F}$ (FOR $G2$ AND $G3$, WE ASSUMED $k = 50$). *LI*, *NE*, AND *NI* REPRESENT THE NUMBER OF LINEAR INEQUALITIES, AND NONLINEAR EQUATIONS AND INEQUALITIES, RESPECTIVELY

Function	n	Type of f	ρ	LI	NE	NI	a
$G1$	13	quadratic	0.0111%	9	0	0	6
$G2$	k	nonlinear	99.8474%	0	0	2	1
$G3$	k	polynomial	0.0000%	0	1	0	1
$G4$	5	quadratic	52.1230%	0	0	6	2
$G5$	4	cubic	0.0000%	2	3	0	3
$G6$	2	cubic	0.0066%	0	0	2	2
$G7$	10	quadratic	0.0003%	3	0	5	6
$G8$	2	nonlinear	0.8560%	0	0	2	0
$G9$	7	polynomial	0.5121%	0	0	4	2
$G10$	8	linear	0.0010%	3	0	3	6
$G11$	2	quadratic	0.0000%	0	1	0	1

the optimum quite closely for test cases $G1$ and $G7$ (with $\rho = 0.0111\%$ and $\rho = 0.0003\%$, respectively), whereas most of the methods experienced difficulties for test case $G10$ (with $\rho = 0.0010\%$). Two quadratic functions (test cases $G1$ and $G7$) with a similar number of constraints (nine and eight, respectively) and an identical number (six) of active constraints at the optimum gave a different challenge to most of these methods. Also, several methods were quite sensitive to the presence of a feasible solution in the initial population. Possibly, a more extensive testing of various methods was required.

Not surprisingly, the experimental results of [42] suggested that making an appropriate *a priori* choice of an evolutionary method for a nonlinear optimization problem remained an open question. It seems that more complex properties of the problem (e.g., the characteristic of the evaluation function together with the topology of the feasible region) may constitute quite significant measures of the difficulty of the problem. Also, some additional measures of the problem characteristics due to the constraints might be helpful. However, this kind of information is not generally available. In [42], the authors wrote: "It seems that the most promising approach at this stage of research is experimental, involving the design of a scalable test suite of constrained optimization problems, in which many $[\cdots]$ features could be easily tuned. Then it should be possible to test new methods with respect to the corpus of all available methods."

Clearly, there is a need for a parameterized test-case generator that can be used for analyzing various methods in a systematic way rather than testing them on a few selected test cases; moreover, it is not clear whether the addition of a few extra test cases is of any help.

In this paper, we propose such a test-case generator for constrained parameter optimization techniques. This generator is capable of creating various test cases with different characteristics:

- problems with different value of ρ: the relative size of the feasible region in the search space
- problems with different numbers and types of constraints
- problems with convex or nonconvex evaluation function, possibly with multiple optima
- problems with highly nonconvex constraints consisting of (possibly) disjoint regions.

All of this can be achieved by setting a few parameters that influence different characteristics of the optimization problem. Such a test-case generator should be very useful for analyzing and comparing different constraint-handling techniques.

There have been some attempts to propose a test-case generator for unconstrained parameter optimization [65], [66]. We are also aware of one attempt to do so for constrained cases: in [63], the author proposed a so-called stepping-stone problem defined as

$$\text{maximize} \sum_{i=1}^{n} (x_i/\pi + 1)$$

where $-\pi \leq x_i \leq \pi$ for $i = 1, \cdots, n$ and the following constraints are satisfied:

$$e^{x_i/\pi} + \cos(2x_i) \leq 1, \qquad \text{for } i = 1, \cdots, n.$$

Note that the evaluation function is linear, and that the feasible region is split into 2^n disjoint parts (called stepping stones). As the number of dimensions n grows, the problem becomes more complex. However, as the stepping-stones problem has only one parameter, it cannot be used to investigate some aspects of a constraint-handling method. In [34] and [35], we reported on preliminary experiments with a test-case generator.

The paper is organized as follows. The following section describes the proposed test-case generator for constrained parameter optimization techniques. Section III briefly surveys several constraint-handling techniques for numerical optimization problems which have emerged in evolutionary computation techniques in recent years. Section IV discusses the experimental results of one particular constraint-handling technique on a few generated test cases. Section V concludes the paper, and indicates some directions for future research.

II. TEST-CASE GENERATOR

As explained in the Introduction, it is of great importance to have a parameterized generator of test cases for constrained parameter optimization problems. By changing the values of some parameters, it would be possible to investigate the merits/drawbacks (efficiency, cost, etc.) of many constraint-handling methods. Several interesting questions could be addressed, as follows, among many others.

- How does the efficiency of a constraint-handling method change as a function of the number of disjoint components of the feasible part of the search space?
- How does the efficiency of a constraint-handling method change as a function of the ratio between the sizes of the feasible part and the whole search space?

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

- What is the relationship between the number of constraints (or the number of dimensions, for example) of a problem and the computational effort of a method?

In the following part of this section, we describe such a parameterized test-case generator:

$$\mathcal{TCG}(n, w, \lambda, \alpha, \beta, \mu).$$

The meaning of its six parameters is as follows:

- n number of variables of the problem
- w a parameter to control the number of optima in the search space
- λ a parameter to control the number of constraints (inequalities)
- α a parameter to control the connectedness of the feasible search regions
- β a parameter to control the ratio of the feasible to total search space
- μ a parameter to influence the ruggedness of the fitness landscape.

The following subsection explains the general ideas behind the proposed concepts. Section II-B describes some details of the test-case generator $\mathcal{TCG}$, and Section II-C graphs a few landscapes. Section II-D discusses further enhancements incorporated in the $\mathcal{TCG}$, and Section II-E summarizes some of its properties.

A. Preliminaries

The general idea is to divide the search space $\mathcal{S}$ into a number of disjoint subspaces $\mathcal{S}_k$, and to define a unimodal function f_k for every $\mathcal{S}_k$. Thus, the evaluation function G is defined on $\mathcal{S}$ as follows:

$$G(\vec{x}) = f_k(\vec{x}), \qquad \text{iff } \vec{x} \in \mathcal{S}_k.$$

The number of subspaces $\mathcal{S}_k$ corresponds to the total number of local optima of function G.

Each subspace $\mathcal{S}_k$ is divided further into its feasible $\mathcal{F}_k$ and infeasible $\mathcal{I}_k$ parts; it may happen that one of these parts is empty. This division of $\mathcal{S}_k$ is obtained by introducing a double inequality that feasible points must satisfy. The feasible part $\mathcal{F}$ of the whole search space $\mathcal{S}$ is defined then as a union of all $\mathcal{F}_k$'s.

The final issue addressed in the proposed model concerns the relative heights of the local optima of functions f_k. The global optimum is always located in $\mathcal{S}_0$, but the heights of other peaks (i.e., local optima) may determine whether the problem would be easier or harder to solve.

Let us now discuss the connections between the above ideas and the parameters of the test-case generator. The first (integer) parameter n of the $\mathcal{TCG}$ determines the number of variables of the problem ($n \geq 1$). The next (integer) parameter $w \geq 1$ determines the number of local optima in the search space, as the search space $\mathcal{S}$ is divided into w^n disjoint subspaces $\mathcal{S}_k$ ($0 \leq k \leq w^n - 1$), and there is a unique unconstrained optimum in each $\mathcal{S}_k$. The subspaces $\mathcal{S}_k$ are defined in Section II-B; however, the idea is to divide the domain of each of n variables into w disjoint and equal-sized segments of length $1/w$. The parameter w also determines the boundaries of the domains of all variables: for all $1 \leq i \leq n, x_i \in [-(1/2w), 1-(1/2w))$. Thus, the search space $\mathcal{S}$ is defined as an n-dimensional rectangle:[1]

$$\mathcal{S} = \prod_{i=1}^{n} \left[-\frac{1}{2w},\ 1 - \frac{1}{2w} \right)$$

consequently, $|\mathcal{S}| = 1$.

The third parameter λ of the test-case generator is related to the number m of constraints of the problem, as the feasible part $\mathcal{F}$ of the search space $\mathcal{S}$ is defined by means on m double inequalities, called "rings" or "hyperspheres" ($1 \leq m \leq w^n$):

$$r_1^2 \leq c_k(\vec{x}) \leq r_2^2, \qquad k = 0, \cdots, m-1 \tag{2}$$

where $0 \leq r_1 \leq r_2$ and each $c_k(\vec{x})$ is a quadratic function:

$$c_k(\vec{x}) = (x_1 - p_1^k)^2 + \cdots + (x_n - p_n^k)^2$$

where $(p_1^k, \cdots, p_n^k)$ is the center of the kth ring.

These m double inequalities define m feasible parts $\mathcal{F}_k$ of the search space:

$$\vec{x} \in \mathcal{F}_k, \qquad \text{iff } r_1^2 \leq c_k(\vec{x}) \leq r_2^2$$

and the overall feasible search space $\mathcal{F} = \cup_{k=0}^{m-1} \mathcal{F}_k$. Note that the interpretation of constraints here is different from the one in the standard definition of the NLP problem [see (1)]: here, the search space $\mathcal{F}$ is defined as a *union* (not intersection) of all double constraints. In other words, a point $\vec{x}$ is feasible if and only if there exists an index $0 \leq k \leq m-1$ such that the double inequality in (2) is satisfied. Note also that, if $m < w^n$, then $\mathcal{F}_k$ are empty for all $m \leq k \leq w^n - 1$.

The parameter $0 \leq \lambda \leq 1$ determines the number m of constraints as follows:

$$m = \lfloor \lambda(w^n - 1) + 1 \rfloor. \tag{3}$$

Clearly, $\lambda = 0$ and $\lambda = 1$ imply $m = 1$ and $m = w^n$, i.e., the minimum and maximum number of constraints, respectively.

There is one important implementational issue connected with a representation of the number w^n. This number might be too large to store as an integer variable in a program (e.g., for $w = 10$ and $n = 500$); consequently, the value of m might be too large as well. Equation (3) should therefore be interpreted as the expected value of random variable m rather than the exact formula. This is achieved as follows. Note that an index k of a subspace $\mathcal{S}_k$ can be represented in an n-dimensional w-ary alphabet (for $w > 1$) as $(q_{1,k}, \cdots, q_{n,k})$, i.e.,

$$k = \sum_{i=1}^{n} q_{i,k} w^{n-i}.$$

Then, for each dimension, we (randomly) select a fraction $(\lambda)^{1/n}$ of indexes; the subspace $\mathcal{S}_k$ contains one of m constraints iff an index $q_{i,k}$ was selected for dimension i ($1 \leq i \leq n$). Note that the probability of selecting any subspace $\mathcal{S}_k$ is λ. For the same reason, later in the paper (see Sections II-B and II-E), new parameters (k' and k'') replace the original k [e.g., (21) and (28)]. The center $(p_1^k, \cdots, p_n^k)$ of

[1] In Section II-D, we define an additional transformation from $[-(1/2w), 1-(1/2w))$ to $[0, 1)$ to simplify the usability of the $\mathcal{TCG}$.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

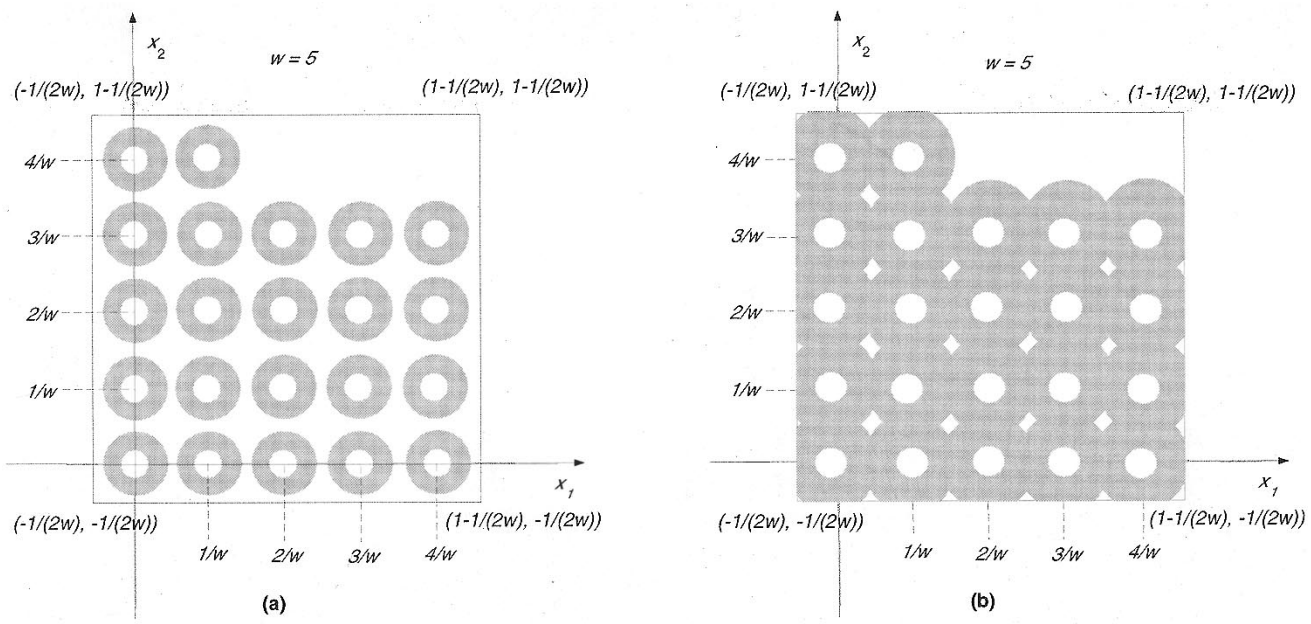

Fig. 1. Search spaces $\mathcal{S}$ and their feasible parts $\mathcal{F}$ (shaded areas) for test cases with $n = 2$, $m = 22$, $r_1 = 0.04$, and (a) $r_2 = 0.09$ or (b) $r_2 = 0.12$.

a hypersphere defined by a particular c_k $(0 \leq k \leq m-1)$ is determined as follows:

$$(p_1^k, \cdots, p_n^k) = (q_{n,k}/w, \cdots, q_{1,k}/w) \tag{4}$$

where $(q_{1,k}, \cdots, q_{n,k})$ is an n-dimensional representation of the number k in w-ary alphabet.[2]

Let us illustrate concepts introduced so far by the following example. Assume that $n = 2$, $w = 5$, and $m = 22$ (note that $m \leq w^n$). Let us assume that r_1 and r_2 [smaller and larger radii of all hyperspheres (circles in this case), respectively] have the following values:[3]

$$r_1 = 0.04 \quad \text{and} \quad r_2 = 0.09.$$

Thus, the feasible search space consists of $m = 22$ disjoint rings, and the ratio ρ between the sizes of the feasible part $\mathcal{F}$ and the whole search space $\mathcal{S}$ is 0.449. The search space $\mathcal{S}$ and the feasible part $\mathcal{F}$ are displayed in Fig. 1(a).

Note that the center of the "first" sphere defined by c_0 (i.e., $k = 0$) is $(p_1^0, p_2^0) = (0, 0)$, as $k = 0 = (0, 0)_5$. Similarly, the center of the "14th" sphere (out of $m = 22$) defined by c_{13} (i.e., $k = 13$) is $(p_1^{13}, p_2^{13}) = (3/w, 2/w)$, as $k = 13 = (2, 3)_5$.

With $r_2 = 0$ (which also implies $r_1 = 0$), the feasible search space would consist of a set of m points. It is interesting to note that, for $r_1 = 0$,

- if $0 \leq r_2 < 1/2w$, the feasible search space $\mathcal{F}$ consists of m disjoint convex components; however the overall search space is nonconvex
- if $1/2w \leq r_2 < \sqrt{n}/2w$, the feasible search space $\mathcal{F}$ consists of one connected component; however, this component is highly nonconvex with many "holes" among spheres [see Fig. 2(a)]
- if $r_2 = \sqrt{n}/2w$, most of the whole search space would be feasible [except the "right top corner" of the search space due to the fact that the number of spheres m might be smaller than w^n; see Fig. 2(b)]. In any case, $\mathcal{F}$ consists of a single connected component.[4]

Based on the above discussions, we can assume that radii r_1 and r_2 satisfy the following inequalities:

$$0 \leq r_1 \leq r_2 \leq \frac{\sqrt{n}}{2w}. \tag{5}$$

Now, we can define the fourth and fifth control parameters of the test-case generator $\mathcal{TCG}$:

$$\alpha = \frac{2wr_2}{\sqrt{n}} \tag{6}$$

$$\beta = \frac{r_1}{r_2}. \tag{7}$$

The operating range for α and β is $0 \leq \alpha, \beta \leq 1$. In that way, the radii r_1 and r_2 are defined by the parameters of the test-case generator:

$$r_1 = \frac{\alpha\beta\sqrt{n}}{2w}, \quad r_2 = \frac{\alpha\sqrt{n}}{2w}. \tag{8}$$

Note that, if $\alpha < 1/\sqrt{n}$, the feasible "islands" $\mathcal{F}_k$ are not connected, as discussed above. On the other hand, for $\alpha \geq 1/\sqrt{n}$, the feasible subspaces $\mathcal{F}_k$ are connected. Thus, the parameter α controls the connectedness of the feasible search space.[5]

For $\beta = 0$, the feasible parts $\mathcal{F}_k$ are convex, and for $\beta > 0$, they are always nonconvex [see (a) and (b) of Fig. 1]. If $\beta = 1$

[2]We assumed here that $w > 1$. For $w = 1$, there is no distinction between the search space $\mathcal{S}$ and the subspace $\mathcal{S}_0$, and consequently, m is 1 (a single double constraint). In the latter case, the center of the only hypersphere is the center of the search space.

[3]These values are determined by two other parameters of the test-case generator, α and β, as discussed later.

[4]Note that the exact value of ρ depends on m: if $m = w^n$, then $\rho = 1$; otherwise, $\rho < 1$.

[5]We will see later that this parameter, together with parameter β, also controls the amount of deviation of the constrained maximum solution from the unconstrained maximum solution.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

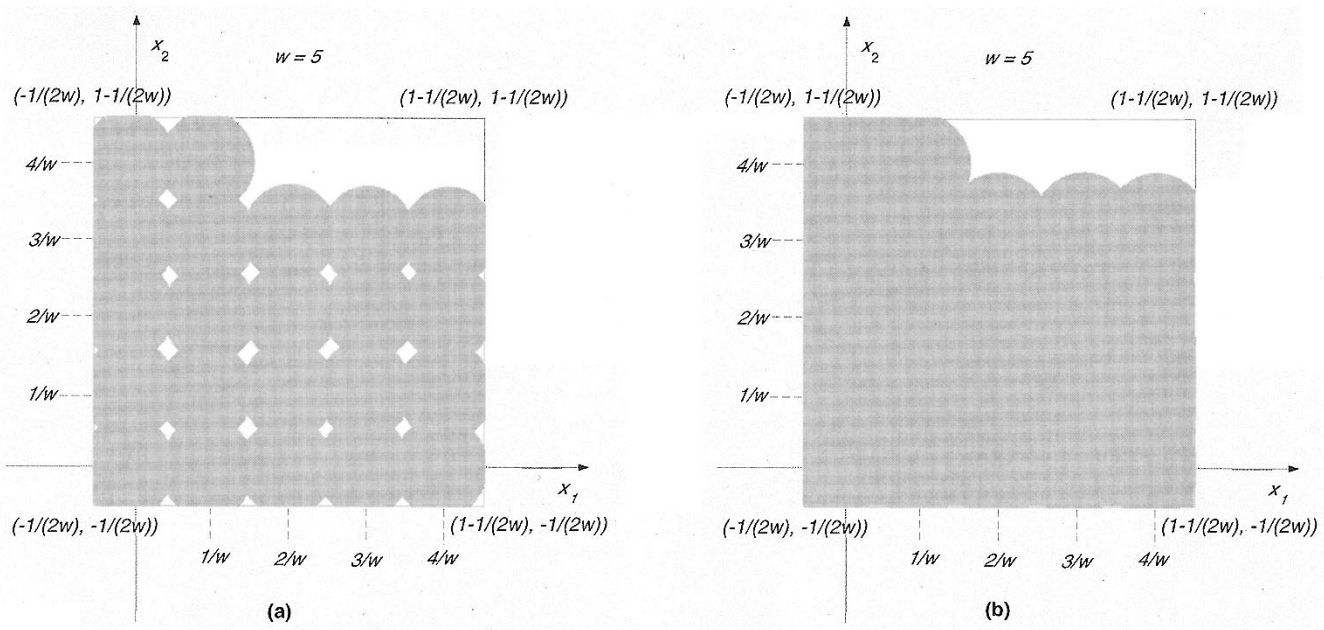

Fig. 2. Search spaces $\mathcal{S}$ and their feasible parts $\mathcal{F}$ (shaded areas) for test cases with $n = 2, m = 22, r_1 = 0$, and (a) $r_2 = 0.12$ or (b) $r_2 = \sqrt{2}/2w = \sqrt{2}/10$.

(i.e., $r_1 = r_2$), the feasible search space consists of (possibly parts of) boundaries of spheres; this case corresponds to an optimization problem with equality constraints. The parameter β is also related to the proportion of the ratio ρ of the feasible to the total search space ($\rho = |\mathcal{F}|/|\mathcal{S}|$). For an n-dimensional hypersphere, the enclosed volume is given as follows [55]:

$$V_n(r) = \frac{\pi^{n/2} r^n}{\Gamma(n/2+1)}. \tag{9}$$

When $\alpha \leq 1/\sqrt{n}$, the ratio ρ can be written as follows[6] (note that $|\mathcal{S}| = 1$):

$$\begin{aligned}\rho &= m(V_n(r_2) - V_n(r_1)) \\ &= \frac{(\pi n)^{n/2}\alpha^n}{2^n \Gamma(n/2+1)} \frac{m}{w^n} (1-\beta^n).\end{aligned} \tag{10}$$

When $\beta \approx 1$ ($\mathcal{F}_k$'s are rings of a small width), the above ratio is

$$\rho \approx \frac{(\pi n)^{n/2}\alpha^n}{2^n \Gamma(n/2+1)} \frac{m}{w^n} n(1-\beta). \tag{11}$$

In the following subsection, we define the test-case generator $\mathcal{TCG}$ in detail; we provide definitions of functions f_k, and discuss the sixth parameter μ of this test-case generator.

B. Details

As mentioned earlier, the search space

$$\mathcal{S} = \prod_{i=1}^{n} \left[-\frac{1}{2w}, 1 - \frac{1}{2w} \right)$$

is divided into w^n subspaces $\mathcal{S}_k$, $k = 0, 1, \cdots, (w^n - 1)$. Each subspace $\mathcal{S}_k$ is defined as an n-dimensional cube: $\mathcal{S}_k = \{x_i\colon l_i^k \leq x_i < u_i^k\}$, where the bounds l_i^k and u_i^k are defined as

$$l_i^k = \frac{2q_{n-i+1,k} - 1}{2w} \quad \text{and} \quad u_i^k = \frac{2q_{n-i+1,k} + 1}{2w} \tag{12}$$

for $k = 0, 1, \cdots, (w^n - 1)$. The parameter $q_{n-i+1,k}$ is the $(n - i + 1)$th component of an n-dimensional representation of the number k in w-ary alphabet. For example, the boundaries of the subspace $\mathcal{S}_{13}$ (see Fig. 1) are

$$\frac{5}{2w} \leq x_1 < \frac{7}{2w} \quad \text{and} \quad \frac{3}{2w} \leq x_2 < \frac{5}{2w}$$

as $(2,3)_5 = 13$, i.e., $(2,3)$ is a two-dimensional representation of $k = 13$ in $w = 5$-ary alphabet.

For each subspace $\mathcal{S}_k$, there is a function f_k defined on this subspace as follows:

$$f_k(x_1, \cdots, x_n) = a_k \left(\prod_{i=1}^{n} \left(u_i^k - x_i\right)\left(x_i - l_i^k\right) \right)^{1/n} \tag{13}$$

where a_k's are predefined positive constants. Note that, for any $\vec{x} \in \mathcal{S}_k$, $f_k(\vec{x}) \geq 0$; moreover, $f_k(\vec{x}) = 0$ iff $\vec{x}$ is a boundary point of the subspace $\mathcal{S}_k$.

The evaluation function (to be maximized) of the test-case generator $\mathcal{TCG}$ is defined as follows:

$$G(x_1, \cdots, x_n) = \begin{cases} f_0(x_1, \cdots, x_n), & \text{if } (x_1, \cdots, x_n) \in \mathcal{S}_0 \\ f_1(x_1, \cdots, x_n), & \text{if } (x_1, \cdots, x_n) \in \mathcal{S}_1 \\ \quad\vdots & \\ f_{w^n-1}(x_1, \cdots, x_n), & \text{if } (x_1, \cdots, x_n) \in \mathcal{S}_{w^n-1} \end{cases} \tag{14}$$

where

- $-(1/2w) \leq x_i < 1 - (1/2w)$ for all $1 \leq i \leq n$, i.e., $\vec{x} = (x_1, \cdots, x_n) \in \mathcal{S}$

[6]For $\alpha > 1$, the hyperspheres overlap, and it is not trivial to compute this ratio. However, this ratio gets closer to 1 as α increases from 1.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

- $\mathcal{F} = \mathcal{F}_0 \cup \cdots \cup \mathcal{F}_{w^n-1}$, i.e., one of the following m double constraints is satisfied:

$$r_1^2 \le c_k \le r_2^2 \qquad (0 \le k \le m-1).$$

Now, we are ready to discuss the significance of the predefined constants a_k (13). Let us introduce the following notation:

- $(\underline{x}_1^k, \cdots, \underline{x}_n^k)$—is the maximum solution for *unconstrained* function f_k, i.e., when the constraint $r_1^2 \le c_k(\vec{x}) \le r_2^2$ is ignored (with $r_1 > 0$)
- $(\overline{x}_1^k, \cdots, \overline{x}_n^k)$—is the maximum solution for *constrained* function f_k, i.e., when the constraint $r_1^2 \le c_k(\vec{x}) \le r_2^2$ is taken into account.

The function f_k has its unconstrained maximum at the center of $\mathcal{S}_k$, i.e.,

$$\underline{x}_i^k = (l_i^k + u_i^k)/2.$$

The corresponding function value at this point is

$$f_k(\underline{x}^k) = \frac{a_k}{4w^2}. \tag{15}$$

Thus, if constraints (with $r_1 > 0$) are not taken into account, the function G has exactly w^n local maxima points: one maximum for each subspace $\mathcal{S}_k$. The global maximum (again, without considering constraints) lies in the subspace $\mathcal{S}_k$, for which the corresponding function f_k has the largest constant a_k [see (13)].

When constraints are taken into account (with $r_1 > 0$), the unconstrained maximum is not feasible anymore. By using first- and second-order optimality conditions [8], it can be shown that the new (feasible) maximum at each subspace moves to

$$\overline{x}_i^k = (l_i^k + u_i^k)/2 \pm r_1/\sqrt{n}$$

which is located on the inner ring. The corresponding maximum function value in subspace S_k is then

$$\begin{aligned} f_k(\overline{x}^k) &= a_k \cdot \left(1/(4w^2) - r_1^2/n\right) \\ &= a_k \cdot \left(\frac{1-\alpha^2\beta^2}{4w^2}\right) = \frac{Aa_k}{4w^2} \end{aligned} \tag{16}$$

where $A = 1 - \alpha^2\beta^2$ (the constant A is important in our further discussion). Clearly, for $r_1 > 0$, $A < 1$. For an n-dimensional problem, there is a total of 2^n maxima points in each subspace, each having the same above maximum function value. Thus, there is a total of $(2w)^n$ maxima in the entire search space.

To control the heights of these local maxima, the values of a_k can be arranged in a particular sequence, so that the global maximum solution always occurs in the first subspace (or S_0) and the worst local maximum solution occurs in the subspace S_{w^n-1}. Different sequences of this type result in landscapes of different complexity. This is the role of the sixth parameter μ of the test-case generator $\mathcal{TCG}$: by changing its value, we should be able to change the landscape from difficult to easy.

Let us first consider two possible scenarios, which would serve as two extreme cases (i.e., difficult and easy landscapes, respectively) for the parameter μ. In both of these cases, we assume that $r_1 > 0$ (otherwise, the unconstrained maximum overlaps with the constrained one).

Case 1: It seems that a challenging landscape would have the following feature: the constrained maximum solution is no better than the worst local maximum solution of the unconstrained function. This way, if a constraint-handling optimizer does not work properly to search in the feasible region, one of many local maximum solutions can be found, instead of the global constrained optimum solution. Realizing that the global maximum solution lies in subspace S_0 and the worst local maximum solution lies in subspace S_{w^n-1}, we have the following condition:

$$f_0\left(\overline{x}_1^0, \overline{x}_2^0, \cdots, \overline{x}_n^0\right) \le f_{w^n-1}\left(\underline{x}_1^{w^n-1}, \underline{x}_2^{w^n-1}, \cdots, \underline{x}_n^{w^n-1}\right). \tag{17}$$

Substituting the function values, we obtain the following:

$$\frac{a_{w^n-1}}{a_0} \ge A. \tag{18}$$

Case 2: On the other hand, let us consider an easy landscape, where the constrained maximum solution is no worse than the unconstrained local maximum solution of the next-best subspace. This makes the function easy to optimize because the purpose is served even if the constraint optimizer is incapable of distinguishing feasible and infeasible regions in the search space, as long as it can find the globally best subspace in the entire search space, and concentrate its search there. Note that, even in this case, there are many local maximum solutions where the optimizer can get stuck. Thus, this condition tests more an optimizer's ability to find the global maximum solution among many local maximum solutions, and does not test as much whether the optimizer can distinguish feasible from infeasible search regions. Mathematically, the following condition must be true:

$$f_0\left(\overline{x}_1^0, \overline{x}_2^0, \cdots, \overline{x}_n^0\right) \ge f_1\left(\underline{x}_1^1, \underline{x}_2^1, \cdots, \underline{x}_n^1\right) \tag{19}$$

assuming that the next-best subspace is $\mathcal{S}_1$. Substituting the function values, we obtain

$$\frac{a_1}{a_0} \le A. \tag{20}$$

Thus, to control the degree of difficulty in the function, i.e., to vary between Cases 1 and 2, the parameter μ is introduced. It can be adjusted at different values to have the above two conditions as extreme cases. To simplify the matter, we may like to have condition (18) when μ is set to 1, and to have condition (20) when μ is set to 0. One such possibility is to have the following term for defining a_k:[7]

$$a_k = (1-\alpha^2\beta^2)^{(1-\mu')k'} = A^{(1-\mu')k'}. \tag{21}$$

The value of k' is given by the following formula:

$$k' = \log_2\left(\sum_{i=1}^{n} q_{i,k} + 1\right) \tag{22}$$

[7]Although this term can be used for any $w \ge 1$, we realize that, for $w = 1$, there is only one subspace $\mathcal{S}_0$, and a_0 is 1, irrespective of μ. For completeness, we set $\mu = 1$ in this case.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

where $(q_{1,k}, \cdots, q_{n,k})$ is an n-dimensional representation of the number k in a w-ary alphabet. The value of μ' is defined as

$$\mu' = \left(1 - \frac{1}{\log_2(nw - n + 1)}\right)\mu \tag{23}$$

for $\mu \in [0, 1]$. The term from (21) has the following properties.

1) All a_k are nonnegative.
2) The value of a_0 for the first subspace S_0 is always 1 as $a_0 = A^0 = 1$.
3) The sequence a_k takes $n(w-1)+1$ different values (for $k = 0, \cdots, w^n - 1$).
4) For $\mu = 0$, the sequence $a_k = A^{k'}$ lies in $[A^{\log_2(nw-n+1)}, 1]$ (with $a_0 = 1$ and $a_{w^n-1} = A^{\log_2(nw-n+1)}$). Note that $a_1 = A^1 = A$, so condition (20) is satisfied. Thus, setting $\mu' = 0$ makes the test function much easier, as discussed above.
5) For $\mu = 1$, the sequence $a_k = A^{k'/\log_2(nw-n+1)}$ lies in the range $[A, 1]$ (with $a_0 = 1$ and $a_{w^n-1} = A$), and condition (18) is satisfied with the equality sign. Thus, we have Case 1 established, which is the most difficult test function of this generator, as discussed above. Since $\mu = 0$ and $\mu = 1$ are two extreme cases of easiness and difficulty, test functions with various levels of difficulty can be generated by using $\mu \in [0, 1]$. Values of μ closer to 1 will create more difficult test functions than values closer to 0. In the next section, we show the effect of varying μ in some two-dimensional test functions.

To make the global constrained maximum function value always at 1, we normalize the a_k constants as follows:

$$\begin{aligned} a'_k &= \frac{a_k}{(1/(4w^2) - r_1^2/n)} = \left(\frac{4w^2}{1 - \alpha^2\beta^2}\right) \cdot a_k \\ &= 4w^2(1 - \alpha^2\beta^2)^{k'(1-\mu')-1} \end{aligned} \tag{24}$$

where k' and μ' are defined by (22) and (23), respectively. Thus, the proposed test-function generator is defined in (14), where the functions f_k are defined in (13). The term a_k in (13) is replaced by a'_k defined in (24). The global maximum solution is at $x_i = \pm r_1/\sqrt{n} = \pm\alpha\beta/(2w)$ for all $i = 1, 2, \cdots, n$, and the function values at these solutions are always equal to 1. It is important to note that there is a total of 2^n global maximum solutions having the same function value of 1, and at all of these maxima, only one constraint $(c_0(\vec{x}) \geq r_1^2)$ is active.

Let us emphasize again that the above discussion is relevant only for $r_1 > 0$. If $r_1 = 0$, then $A = 1 - \alpha^2\beta^2 = 1$, and all a_k's are equal to 1. Thus, we need a term defining a_k if $\alpha\beta = 0$ to make the peaks of variable heights. In this case, we can set

$$a_k = \frac{(\mu - 1)k''}{n(w-1)} + 1. \tag{25}$$

The value of k'' is given by the following formula:

$$k'' = \sum_{i=1}^{n} q_{i,k}$$

where $(q_{1,k}, \cdots, q_{n,k})$ is an n-dimensional representation of the number k in w-ary alphabet. Thus, for $\mu = 0$, we have $a_0 = 1$ and $a_{w^n-1} = 0$ (the steepest decline of heights of the peaks), whereas for $\mu = 1$, $a_0 = a_1 = \cdots = a_{w^n-1} = 1$ (no decline). To make sure that the global maximum value of the constrained function is equal to 1, we can set

$$a'_k = 4w^2 a_k \tag{26}$$

for all k.

C. A Few Examples

In order to obtain a feel for the evaluation function at this stage of building the $\mathcal{TCG}$, we plot the surface for different values of control parameters, and with $n = 2$ (two variables only). A very simple unimodal test problem can be generated by using the following parameter setting: $\mathcal{TCG}(2, 1, 0, (1/\sqrt{2}), 0, 1)$ (see Fig. 3). Note that, in this case, $\beta = 0$, which implies that $r_1 = 0$ [see (21)]. To show the reducing peak effect, we need $w > 1$ and $\beta > 0$ as, for $\beta = 0$, all peaks have the same height. Thus, we assume that $\beta = 0.8$, $w = 4$, and $\lambda = 1$ (i.e., $m = w^n$) in all remaining test cases depicted in the figures of this section.

In Fig. 4, we show $\mathcal{TCG}(2, 4, 1, (1/\sqrt{2}), 0.8, 1.0)$, thereby having $r_2 = 0.125$ and $r_1 = 0.1$. Using (10), we obtain the ratio ρ of feasible to total search space as 0.1875π or 0.589.

Let us discuss the effect of the parameter μ on the shape of the landscape. Figs. 4–6 are plotted with $w = 4$, but different μ values are used. Recall that the optimal solution always lies in the island closest to the origin. As the μ value is decreased, the feasible island containing the global optimum solution becomes more emphasized. The problem depicted in Fig. 4 will be more difficult to optimize for a global solution than that in Fig. 6 because, in Fig. 4, other feasible islands also contain comparably good solutions.

Note, again, that for a nonzero r_1, the optimal solutions move to the inner boundary (exactly at $x_i = \pm\alpha\beta/(2w)$. Consequently, these problems are harder for optimization algorithms than for problems with $r_1 = 0$. A nice aspect of these test functions is that the global feasible function value is always 1, no matter what control parameter values are chosen. Fig. 7 shows a slice of the function at $x_2 = 0$ for a test problem $\mathcal{TCG}(2, 4, 1, (1/\sqrt{2}), 0.8, 1.0)$, thus having $r_2 = 0.125$ and $r_1 = 0.1$. Function values corresponding to infeasible solutions are marked using dashed lines. The point "P" is the globally optimal function value. The infeasible function values (like point "Q") higher than 1 are also shown. If the constraint-handling strategy is not proper, an optimization algorithm may get trapped in local optimal solutions, like "Q," which has a better function value, but is infeasible.

D. Further Enhancements

As explained in Section II-A, the general idea behind the test-case generator $\mathcal{TCG}$ was to divide the search space $\mathcal{S}$ into a number of disjoint subspaces $\mathcal{S}_k$, and to define a unimodal function f_k for every $\mathcal{S}_k$. The number of subspaces $\mathcal{S}_k$ corresponds to the total number of local optima of function G. To control the heights of these local maxima, the values of a_k were arranged in a particular sequence, so that the global maximum solution always occurs in the first subspace (for $k = 0$, i.e., in S_0), and the worst local maximum solution occurs in the last subspace

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

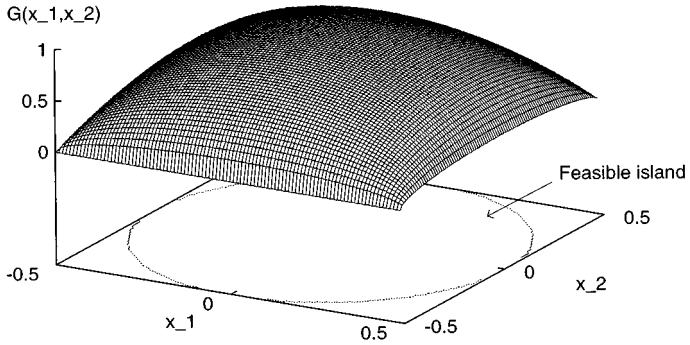

Fig. 3. Test case $\mathcal{TCG}(2, 1, 0, 1/\sqrt{2}, 0, 1)$.

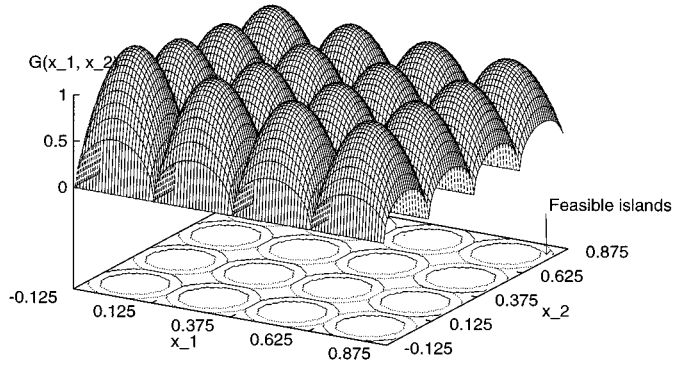

Fig. 4. Test case $\mathcal{TCG}(2, 4, 1, 1/\sqrt{2}, 0.8, 1)$.

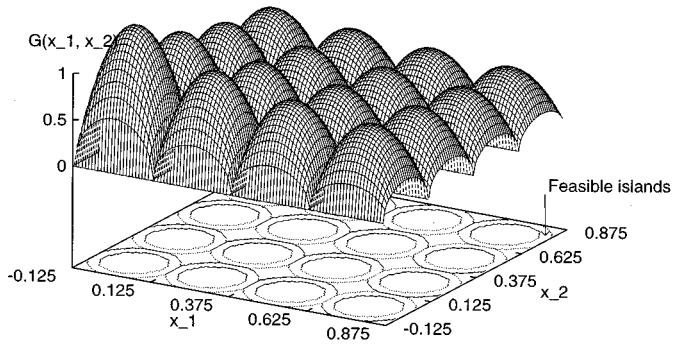

Fig. 5. Test case $\mathcal{TCG}(2, 4, 1, 1/\sqrt{2}, 0.8, 0.5)$.

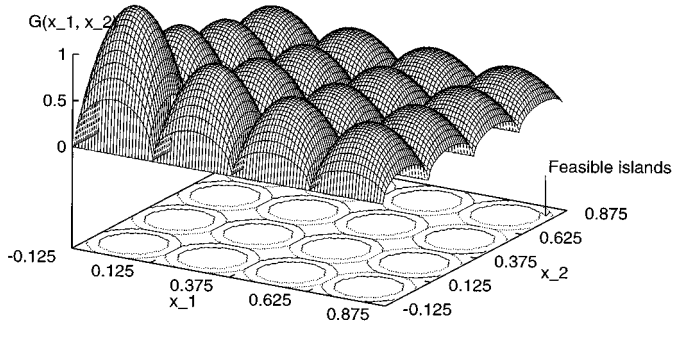

Fig. 6. Test case $\mathcal{TCG}(2, 4, 1, 1/\sqrt{2}, 0.8, 0)$.

(for $k = w^n - 1$, i.e., in S_{w^n-1}). Note also that subspaces $\mathcal{S}_k$ of the test-case generator $\mathcal{TCG}$ are arranged in a particular order. For example, for $n = 2$ and $w = 4$, the highest peak is located in subspace $\mathcal{S}_0$, two next-highest peaks (of the same height) are

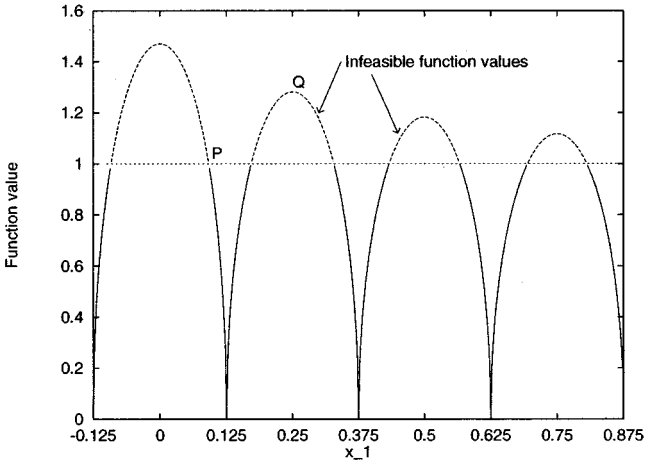

Fig. 7. A slice through $x_2 = 0$ of a test problem with $w = 4$, $\mu = 1.0$, and $r_1 = 0.1$ is shown. The global optimum solution is in the leftmost attractor.

in subspaces $\mathcal{S}_1$ and $\mathcal{S}_4$, three next-highest peaks are in $\mathcal{S}_2$, $\mathcal{S}_5$, $\mathcal{S}_8$, etc. (see Figs. 4–6).

To remove this fixed pattern from the generated test cases, we need an additional mechanism: a random permutation of subspaces $\mathcal{S}_k$. Thus, we need a procedure *Transform* that randomly permutes indexes of subspaces. This\bye is any function

$$Transform: \quad [0 \cdots N] \rightarrow [0 \cdots N]$$

such that, for any $0 \leq p \neq q \leq N$, $Transform(p) \neq Transform(q)$. The advantage of such a procedure *Transform* is that we do not need any memory for storing the mapping between indexes of subspaces (the size of such memory, for $w > 1$ and large n, may be excessive); however, there is a need to recompute the permuted index of a subspace $\mathcal{S}_k$ every time we need it.

The procedure *Transform* also can be used to generate a different allocation for m constraints. Note that the $\mathcal{TCG}$ allocates m rings (each defined by a pair of constraints) always to the first m subspaces $\mathcal{S}_k, k = 0, 1, \cdots, m-1$ (see, for example, Fig. 1). However, care should be taken to always allocate a ring in the subspace containing the higest peak (thus, we know the value of the global feasible solution).

For example, for $n = 2$ and $w = 4$, our implementation of the procedure *Transform* generated the following permutation of peaks of f_k's:

$$\begin{array}{cccc} 0 \rightarrow 5 & 1 \rightarrow 13 & 2 \rightarrow 9 & 3 \rightarrow 8 \\ 4 \rightarrow 10 & 5 \rightarrow 14 & 6 \rightarrow 15 & 7 \rightarrow 7 \\ 8 \rightarrow 6 & 9 \rightarrow 3 & 10 \rightarrow 0 & 11 \rightarrow 2 \\ 12 \rightarrow 1 & 13 \rightarrow 4 & 14 \rightarrow 12 & 15 \rightarrow 11. \end{array}$$

Fig. 8 shows the final outcome (landscape with transformed peaks; see Fig. 6 for the original landscape) for the test case $\mathcal{TCG}(2, 4, 1, (1/\sqrt{2}), 0.8, 0)$.

The final modification of the $\mathcal{TCG}$ maps the domains of parameters x_i from $[-(1/2w), 1 - (1/2w))$ into $[0, 1)$; thus, the new search space $\mathcal{S}'$ is

$$\mathcal{S}' = \prod_{i=1}^{n} [0, 1)$$

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

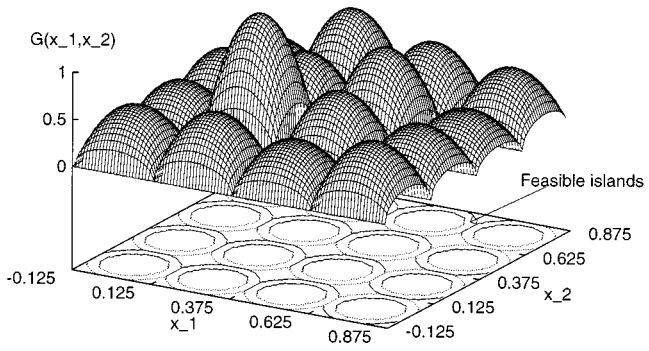

Fig. 8. Transformed peaks for the test case $\mathcal{TCG}(2, 4, 1, 1/\sqrt{2}, 0.8, 0)$.

which is more "user friendly" than the original $\mathcal{S}$. The mapping is a straightforward linear transformation:

$$x'_i \leftarrow x_i + \frac{1}{2w}$$

for all parameters x_i $(i = 1, \cdots, n)$. This means that the user supplies an input vector $\vec{x} \in [0, 1)^n$, whereas $\mathcal{TCG}$ returns the evaluation value and a measure of the constraint violation.

E. Summary

Based on the above presentations, we now summarize the properties of the test-case generator $\mathcal{TCG}(n, w, \lambda, \alpha, \beta, \mu)$. The parameters are

$$\begin{array}{ll} n \geq 1; \ \text{integer} & w \geq 1; \ \text{integer} \\ 0 \leq \lambda \leq 1; \ \text{float} & 0 \leq \alpha \leq 1; \ \text{float} \\ 0 \leq \beta \leq 1; \ \text{float} & 0 \leq \mu \leq 1; \ \text{float}. \end{array}$$

The evaluation function G is defined by formula (14), where each f_k $(k = 0, \cdots, w^n - 1)$ is defined on $\mathcal{S}_k$ as

$$f_k(x'_1, \cdots, x'_n) = a'_k \left(\prod_{i=1}^{n} \left(u_i^k - x_i\right) \left(x_i - l_i^k\right) \right)^{1/n}. \quad (27)$$

All variables x'_i's have the same domain:

$$x'_i \in [0, 1).$$

Note, that $x'_i = x_i + (1/2w)$. The constants a'_k are defined as follows:

$$a'_k = \begin{cases} 4w^2(1 - \alpha^2\beta^2)^{k'[(1-\mu)+\mu/(\log_2(wn-n+1)]-1}, \\ \qquad \text{if } \alpha\beta > 0 \\ 4w^2 \left(\dfrac{(\mu - 1)k''}{n(w-1)} + 1 \right), \\ \qquad \text{if } \alpha\beta = 0 \end{cases} \quad (28)$$

where

$$k' = \log_2 \left(\sum_{i=1}^{n} q_{i,k} + 1 \right) \quad \text{and} \quad k'' = \sum_{i=1}^{n} q_{i,k}$$

where $(q_{1,k}, \cdots, q_{n,k})$ is an n-dimensional representation of the number k in w-ary alphabet.

If $r_1 > 0$ (i.e., $\alpha\beta > 0$), the function G has 2^n global maxima points, all in permuted $\mathcal{S}_0$. For any global solution $(x_1, \cdots, x_n)$, $x_i = \pm\alpha\beta/(2w)$ for all $i = 1, 2, \cdots, n$. The function values at these solutions are always equal to 1.

On the other hand, if $r_1 = 0$ (i.e., $\alpha\beta = 0$), the function G has either one global maximum (if $\mu < 1$) or m maxima points (if $\mu = 1$), one in each of permuted $\mathcal{S}_0, \cdots, \mathcal{S}_{m-1}$. If $\mu < 1$, the global solution $(x_1, \cdots, x_n)$ is always at

$$(x_1, \cdots, x_n) = (0, 0, \cdots, 0).$$

.

The interpretation of the six parameters of the test-case generator $\mathcal{TCG}$ is as follows.

1) *Dimensionality* n: By increasing the parameter n, the dimensionality of the search space can be increased.
2) *Multimodality* w: By increasing the parameter w, the multimodality of the search space can be increased. For the unconstrained function, there are w^n local maximum solutions, of which one is globally maximum. For the constrained test function with $\alpha\beta > 0$, there are $(2w)^n$ different local maximum solutions, of which 2^n are globally maximum solutions.
3) *Number of Constraints* λ: By increasing the parameter λ, the number m of constraints is increased.
4) *Connectedness* α: By reducing the parameter α (from 1 to $1/\sqrt{n}$ and smaller), the connectedness of the feasible subspaces can be reduced. When $\alpha < 1/\sqrt{n}$, the feasible subspaces $\mathcal{F}_k$ are completely disconnected. Additionally, parameter α (with fixed β) influences the proportion of the feasible search space to the complete search space (ratio ρ).
5) *Feasibility* β: By increasing the ratio β, the proportion of the feasible search space to the complete search space can be reduced. For β values closer to 1, the feasible search space becomes smaller and smaller. These test functions can be used to test an optimizer's ability to be find and maintain feasible solutions.
6) *Ruggedness* μ: By increasing the parameter μ, the function's ruggedness can be increased (for $\alpha\beta > 0$). A sufficiently rugged function will test an optimizer's ability to search for the globally constrained maximum solution in the presence of other almost equally significant local maxima.

Increasing each of the above parameters (except α) and decreasing α will cause an increased difficulty for any optimizer. However, it is difficult to conclude which of these factors most profoundly affects the performance of an optimizer. Thus, it is recommended that the user should first test his/her algorithm with the simplest possible combination of the above parameters (small n, small w, small μ, large α, small β, and small λ). Thereafter, the parameters may be changed in a systematic manner to create more difficult test functions. The seemingly most difficult test function is created when large values of parameters n, w, λ, β, and μ together with a small value of parameter α are used.

III. Constraint-Handling Methods

During the last few years, several methods have been proposed for handling constraints by evolutionary algorithms

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

for parameter optimization problems. These methods can be grouped into five categories: 1) methods based on preserving feasibility of solutions, 2) methods based on penalty functions, 3) methods which make a clear distinction between feasible and infeasible solutions, 4) methods based on decoders, and 5) other hybrid methods. We discuss them briefly in turn.

A. Methods Based on Preserving Feasibility of Solutions

The best example of this approach is the Genocop (for genetic algorithm for numerical optimization of constrained problems) system [37], [38]. The idea behind the system is based on specialized operators which transform feasible individuals into feasible individuals, i.e., operators, which are closed on the feasible part $\mathcal{F}$ of the search space. The method assumes linear constraints only and a feasible starting point (or feasible initial population). Linear equations are used to eliminate some variables; they are replaced as a linear combination of remaining variables. Linear inequalities are updated accordingly. A closed set of operators maintains the feasibility of solutions. For example, when a particular component x_i of a solution vector $\vec{x}$ is mutated, the system determines its current domain $\mathrm{dom}(x_i)$ (which is a function of linear constraints and remaining values of the solution vector $\vec{x}$), and the new value of x_i is taken from this domain (either with flat probability distribution for uniform mutation, or other probability distributions for nonuniform and boundary mutations). In any case, the offspring solution vector is always feasible. Similarly, arithmetic crossover $a\vec{x}+(1-a)\vec{y}$ of two feasible solution vectors $\vec{x}$ and $\vec{y}$ always yields a feasible solution (for $0 \leq a \leq 1$) in convex search spaces (the system assumes linear constraints only which imply convexity of the feasible search space $\mathcal{F}$).

Recent work [41], [57], [58] on systems that search only the boundary area between feasible and infeasible regions of the search space constitutes another example of the approach based on preserving the feasibility of solutions. These systems are based on specialized boundary operators (e.g., sphere crossover, geometrical crossover, etc.): it is a common situation for many constrained optimization problems that some constraints are active at the target global optimum, thus, the optimum lies on the boundary of the feasible space.

B. Methods Based on Penalty Functions

Many evolutionary algorithms incorporate a constraint-handling method based on the concept of (exterior) penalty functions that penalize infeasible solutions. Usually, the penalty function is based on the distance of a solution from the feasible region $\mathcal{F}$, or on the effort to "repair" the solution, i.e., to force it into $\mathcal{F}$. The former case is the most popular; in many methods, a set of functions f_j $(1 \leq j \leq m)$ is used to construct the penalty, where the function f_j measures the violation of the jth constraint in the following way:

$$f_j(\vec{x}) = \begin{cases} \max\{0, g_j(\vec{x})\}, & \text{if } 1 \leq j \leq q \\ |h_j(\vec{x})|, & \text{if } q+1 \leq j \leq m. \end{cases}$$

However, these methods differ in many important details: how the penalty function is designed and applied to infeasible solutions. For example, a method of static penalties was proposed [19]; it assumes that, for every constraint, we establish a family of intervals which determine the appropriate penalty coefficient. The method of dynamic penalties was examined [20], where individuals are evaluated (at the iteration t) by the following formula:

$$\mathrm{eval}(\vec{x}) = f(\vec{x}) + (C \times t)^{\alpha} \sum_{j=1}^{m} f_j^{\beta}(\vec{x})$$

where C, α, and β are constants. Another approach (Genocop II), also based on dynamic penalties, was described in [32]. In that algorithm, at every iteration, active constraints only are considered, and the pressure on infeasible solutions is increased due to the decreasing values of temperature τ. In [12], a method for solving constraint satisfaction problems that changes the evaluation function based on the performance of an EA run was described: the penalties (weights) of those constraints which are violated by the best individual after termination are raised, and the new weights are used in the next run. A method based on adaptive penalty functions was developed in [3] and [16]: one component of the penalty function takes a feedback from the search process. Each individual is evaluated by the formula

$$\mathrm{eval}(\vec{x}) = f(\vec{x}) + \lambda(t) \sum_{j=1}^{m} f_j^2(\vec{x})$$

where $\lambda(t)$ is updated every generation t with respect to the current state of the search (based on last k generations). The adaptive penalty function was also used in [61], where both the search length and constraint severity feedback were incorporated. It involves the estimation of a near-feasible threshold q_j for each constraint $1 \leq j \leq m$); such thresholds indicate distances from the feasible region $\mathcal{F}$ which are "reasonable" (or, in other words, which determine "interesting" infeasible solutions, i.e., solutions relatively close to the feasible region). An additional method (the so-called segregated genetic algorithm) was proposed in [26] as yet another way to handle the problem of the robustness of the penalty level: two different penalized fitness functions with static penalty terms p_1 and p_2 were designed (smaller and larger, respectively). The main idea is that such an approach will result roughly in maintaining two subpopulations: the individuals selected on the basis of f_1 will more likely lie in the infeasible region, while the ones selected on the basis of f_2 will probably stay in the feasible region. The overall process is thus allowed to reach the feasible optimum from both sides of the boundary of the feasible region.

C. Methods Based on a Search for Feasible Solutions

There are a few methods that emphasize the distinction between feasible and infeasible solutions in the search space $\mathcal{S}$. One method, proposed in [59] (called a "behavioral memory" approach), considers the problem constraints in a sequence; a switch from one constraint to another is made upon arrival of a sufficient number of feasible individuals in the population.

The second method, developed in [49], is based on a classical penalty approach with one notable exception. Each individual is evaluated by the formula

$$\mathrm{eval}(\vec{x}) = f(\vec{x}) + r \sum_{j=1}^{m} f_j(\vec{x}) + \theta(t, \vec{x})$$

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

where r is a constant; however, the original component $\theta(t, \vec{x})$ is an additional iteration-dependent function that influences the evaluations of infeasible solutions. The point is that the method distinguishes between feasible and infeasible individuals by adopting an additional heuristic rule (suggested earlier in [54]): for any feasible individual $\vec{x}$ and any infeasible individual $\vec{y}, \text{eval}(\vec{x}) < \text{eval}(\vec{y})$, i.e., any feasible solution is better than any infeasible one (for minimization problems). In a recent study [9], a modification to this approach was implemented with the tournament selection operator, and with the following evaluation function:

$$\text{eval}(\vec{x}) = \begin{cases} f(\vec{x}), & \text{if } \vec{x} \text{ is feasible} \\ f_{\max} + \sum_{j=1}^{m} f_j(\vec{x}), & \text{otherwise} \end{cases}$$

where $f_{\max}$ is the function value of the worst feasible solution in the population. The main difference between this approach and Powell and Skolnick's approach is that, in this approach, the evaluation function value is not considered in evaluating an infeasible solution. Additionally, a niching scheme is introduced to maintain diversity among feasible solutions. Thus, initially, the search focuses on finding feasible solutions, and later, when an adequate number of feasible solutions is found, the algorithm finds better feasible solutions by maintaining a diversity in solutions in the feasible region. It is interesting to note that there is no need for the penalty coefficient r here because the feasible solutions are always evaluated to be better than infeasible solutions, and infeasible solutions are compared purely based on their constraint violations. However, the normalization of constraints $f_j(\vec{x})$ is suggested. In a number of test problems and in an engineering design problem, this approach is better able to find constrained optimum solutions than Powell and Skolnick's approach.

The third method (Genocop III), proposed in [40], is based on the idea of repairing infeasible individuals. Genocop III incorporates the original Genocop system, but also extends it by maintaining two separate populations, where a development in one population influences the evaluations of individuals in the other population. The first population P_s consists of so-called search points from $\mathcal{F}_l$ which satisfy linear constraints of the problem. The feasibility (in the sense of linear constraints) of these points is maintained by specialized operators. The second population P_r consists of so-called reference points from $\mathcal{F}$; these points are fully feasible, i.e., they satisfy *all* constraints. Reference points $\vec{r}$ from P_r, being feasible, are evaluated directly by the evaluation function [i.e., $\text{eval}(\vec{r}) = f(\vec{r})$]. On the other hand, search points from P_s are "repaired" for evaluation.

D. Methods Based on Decoders

Decoders offer an interesting option for all practitioners of evolutionary techniques. In these techniques, a chromosome "gives instructions" on how to build a feasible solution. For example, a sequence of items for the knapsack problem can be interpreted as: "take an item if possible"—such an interpretation would lead always to a feasible solution.

However, it is important to point out that several factors should be taken into account while using decoders. Each decoder imposes a mapping M between a feasible solution and a decoded solution. It is important that several conditions are satisfied: 1) for each solution $s \in \mathcal{F}$, there is an encoded solution d; 2) each encoded solution d corresponds to a feasible solution s; and 3) all solutions in $\mathcal{F}$ should be represented by the same number of encodings d.[8] Additionally, it is reasonable to request that: 4) the mapping M is computationally fast, and 5) it has a locality feature in the sense that small changes in the coded solution result in small changes in the solution itself. An interesting study on coding trees in genetic algorithm was reported in [46], where the above conditions were formulated.

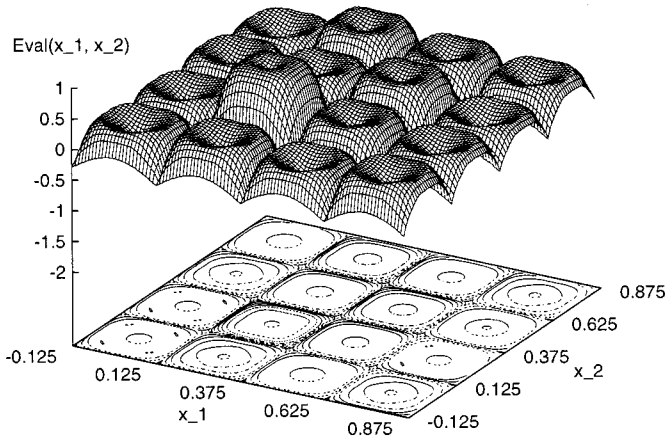

Fig. 9. Final landscape for the test case $\mathcal{TCG}(2, 4, 1, 1/\sqrt{2}, 0.8, 0)$. The penalty coefficient W is set to 1. Contours with function values ranging from -0.2 to 1.0 at a step of 0.2 are drawn at the base.

However, the use of decoders for continuous domains has not been investigated. Only recently [24], [25] has a new approach for solving constrained numerical optimization problems been proposed. This approach incorporates a homomorphous mapping between an n-dimensional cube and a feasible search space. The mapping transforms the constrained problem at hand into an unconstrained one. The method has several advantages over methods proposed earlier (no additional parameters, no need to evaluate—or penalize—infeasible solutions, easiness of approaching a solution located on the edge of the feasible region, no need for special operators, etc.).

E. Hybrid Methods

It is relatively easy to develop hybrid methods that combine evolutionary computation techniques with deterministic procedures for numerical optimization problems. In [64], the authors combined an evolutionary algorithm with the direction set method of Hooke–Jeeves is described; this hybrid method was tested on three (unconstrained) test functions. In [44], the authors considered a similar approach, but they experimented with constrained problems. Again, they combined the evolutionary algorithm with some other method—developed in [27].

[8]However, as observed by Davis [7], the requirement that all solutions in $\mathcal{F}$ should be represented by the same number of decodings seems overly strong: there are cases in which this requirement might be suboptimal. For example, suppose we have a decoding and an encoding procedure which make it impossible to represent suboptimal solutions, and which encode the optimal one: this might be a good thing. (An example would be a graph-coloring order-based chromosome, with a decoding procedure that gives each node its first legal color. This representation could not encode solutions where some nodes that could be colored were not colored, but this is a good thing!)

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

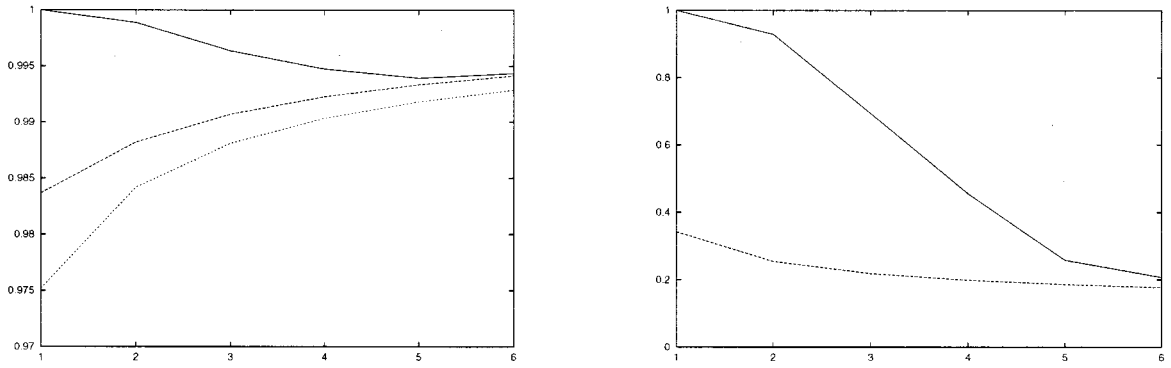

Fig. 10. Average of 100 runs of the system for the $\mathcal{TCG}(n, 10, 1.0, 0.9/\sqrt{n}, 0.1, 0.1)$.

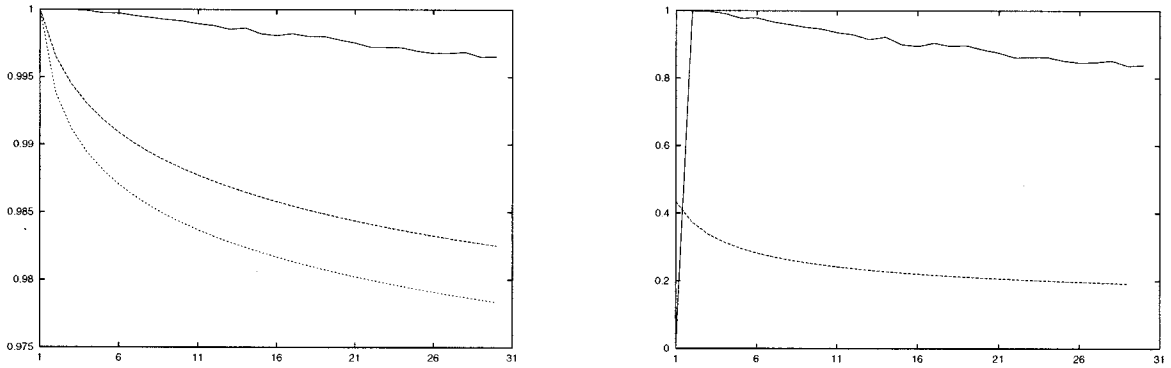

Fig. 11. Average of 100 runs of the system for the $\mathcal{TCG}(2, w, 1.0, 0.9/\sqrt{2}, 0.1, 0.1)$.

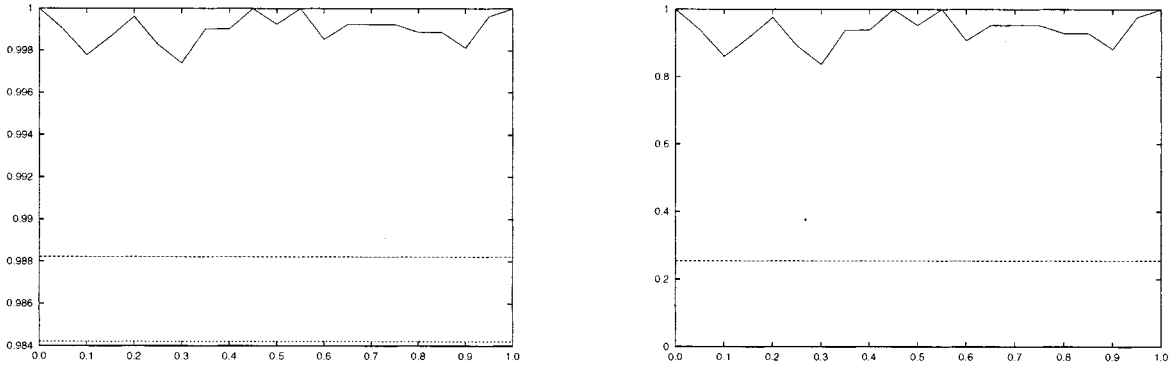

Fig. 12. Average of 100 runs of the system for the $\mathcal{TCG}(2, 10, \lambda, 0.9/\sqrt{2}, 0.1, 0.1)$.

However, while the method of [64] incorporated the direction set algorithm as a problem-specific operator of his evolutionary technique, in [44], the whole optimization process was divided into two separate phases.

Several other constraint-handling methods also deserve some attention. For example, some methods use the values of evaluation function f and penalties f_j $(j = 1, \cdots, m)$ as elements of a vector, and apply multiobjective techniques to minimize all components of the vector. For example, in [56], the vector-evaluated genetic algorithm (VEGA) selects $1/(m+1)$ of the population based on each of the objectives. Such an approach was incorporated by Parmee and Purchase [48] in the development of techniques for constrained design spaces. On the other hand, in the approach of [62], all members of the population are ranked on the basis of constraint violation. Such rank r, together with the value of the evaluation function f, leads to the two-objective optimization problem. This approach gave a good performance on the optimization of gas supply networks.

Also, an interesting approach was reported in [47]. The method (described in the context of constraint satisfaction problems) is based on a coevolutionary model, where a population of potential solutions coevolves with a population of constraints: fitter solutions satisfy more constraints, whereas fitter constraints are violated by fewer solutions. There is some development connected with generalizing the concept of "ant colonies" [5] (which were originally proposed for order-based problems) to numerical domains [4]; the first experiments on some test problems gave very good results [67]. It is also

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

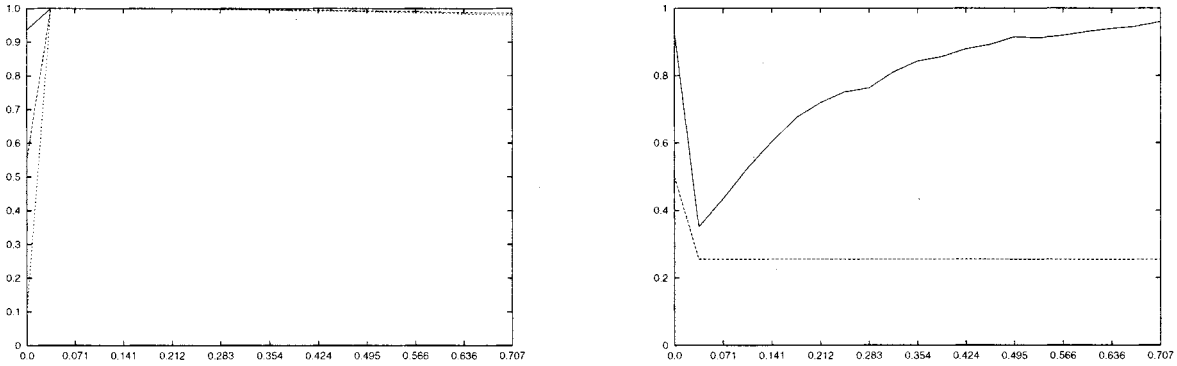

Fig. 13. Average of 100 runs of the system for the $\mathcal{TCG}(2, 10, 1.0, \alpha, 0.1, 0.1)$.

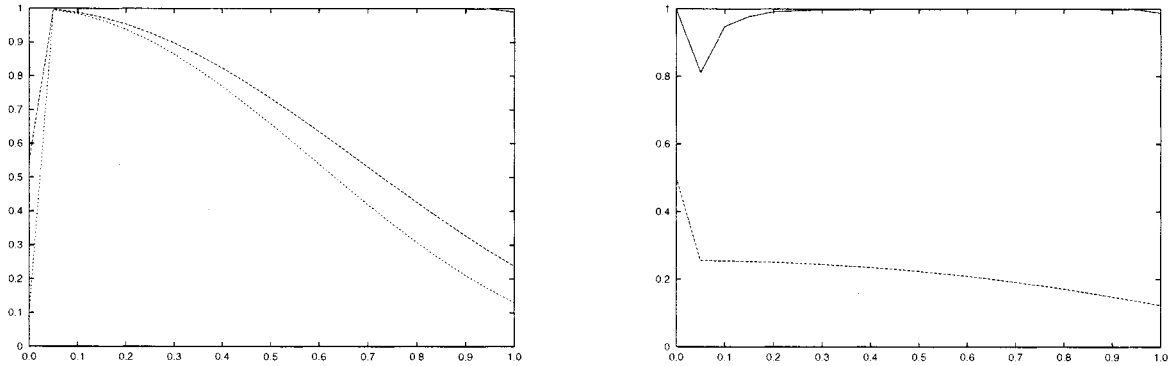

Fig. 14. Average of 100 runs of the system for the $\mathcal{TCG}(2, 10, 1.0, 0.9/\sqrt{2}, \beta, 0.1)$.

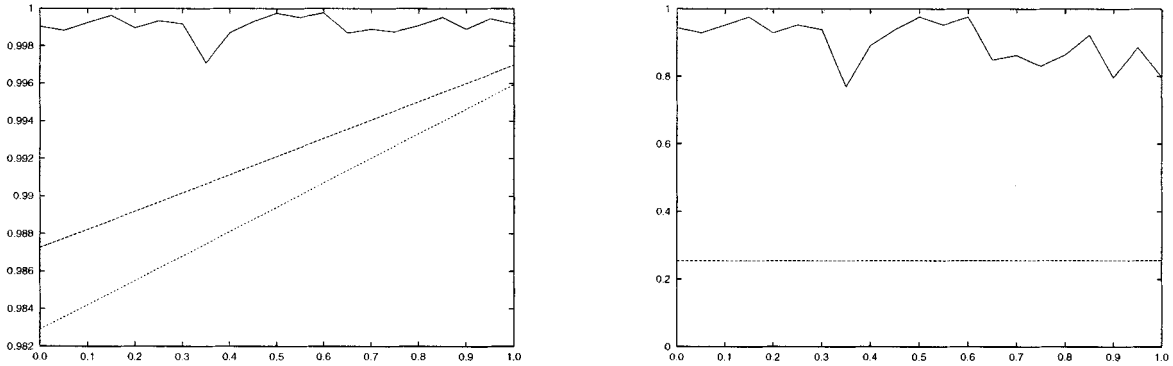

Fig. 15. Average of 100 runs of the system for the $\mathcal{TCG}(2, 10, 1.0, 0.9/\sqrt{2}, 0.1, \mu)$.

possible to incorporate the knowledge of the constraints of the problem into the belief space of cultural algorithms [52]; such algorithms provide the possibility of conducting an efficient search of the feasible search space [53].

IV. Experimental Results

One of the simplest and the most popular constraint-handling methods is based on static penalties. In the following, we define a simple static penalty method, and investigate its properties using the $\mathcal{TCG}$.

The investigated static penalty method is defined as follows. For a maximization problem with the evaluation function f and m constraints,

$$\text{eval}(\vec{x}) = f(\vec{x}) - W \cdot v(\vec{x})$$

where $W \geq 0$ is a constant (penalty coefficient), and function v measures the constraint violation. (Note that only one double constraint is taken into account as the $\mathcal{TCG}$ defines the feasible part of the search space as a union of all m double constraints.)

The evaluation function f is defined by the test-case generator [see definition of function G; (14)]. The constraint violation value of v for any $\vec{x}$ is defined by the following procedure:

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

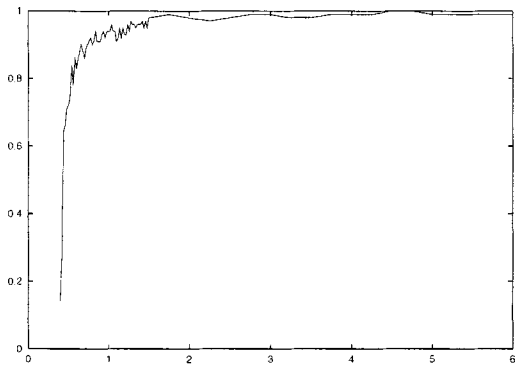

Fig. 16. Success rate of the system for different values of penalty coefficient W. The success rate gives the fraction of the final solutions (out of 250 runs) that were feasible.

```
find k such that x⃗ ∈ S_k
set C = (2w)/√n
if the whole S_k is infeasible then v(x⃗) = 1
else begin
   calculate distance Dist between x⃗
   and the center of the subspace S_k
   if Dist < r_1 then v(x⃗) = C · (r_1 − Dist)
   else if Dist > r_2 then v(x⃗) = C · (Dist − r_2)
   else v(x⃗) = 0
  end
```

The radii r_1 and r_2 are defined by (8). Thus, the constraint violation measure v returns

- 1 if the evaluated point is in infeasible subspace (i.e., subspace without a ring)
- 0 if the evaluated point is feasible
- q if the evaluated point is infeasible, but the corresponding subspace is partially feasible.

This means that the point $\vec{x}$ is either inside the smaller ring or outside the larger one. In both cases, q is a scaled distance of this point to the boundary of a closer ring. Note that the scaling factor C guarantees that $0 < q \leq 1$.

Note that the values of the evaluation function for feasible points of the search space stay in the range [0, 1]; the value at the global optimum is always 1. Thus, both the function and constraint violation values are normalized in [0, 1]. Fig. 9 displays the final landscape for test case $\mathcal{TCG}(2, 4, 1, 1/\sqrt{2}, 0.8, 0)$ (Fig. 8 displays the landscape before penalties are applied).

To test the usefulness of the $\mathcal{TCG}$, a simple steady-state evolutionary algorithm was developed. We have used a constant population size of 100, and each individual is a vector $\vec{x}$ of n floating-point components. Parent selection was performed by a standard binary tournament selection. An offspring replaces the worst individual determined by a binary tournament. One of three operators was used in every generation (the selection of an operator was done according to constant probabilities 0.5, 0.15, and 0.35, respectively):

- Gaussian mutation: $\vec{x} \leftarrow \vec{x} + N(0, \vec{\sigma})$, where $N(0, \vec{\sigma})$ is a vector of independent random Gaussian numbers with a mean of zero and standard deviations $\vec{\sigma}$ (in all experiments reported in this section, we have used a value of $1/(2\sqrt{n})$, which depends only on the dimensionality of the problem)
- uniform crossover: $\vec{z} \leftarrow (z_1, \cdots, z_n)$, where each z_i is either x_i or y_i (with equal probabilities), where $\vec{x}$ and $\vec{y}$ are two selected parents
- heuristic crossover: $\vec{z} = r \cdot (\vec{x} - \vec{y}) + \vec{x}$, where r is a uniform random number between 0 and 0.25, and the parent $\vec{x}$ is not worse than $\vec{y}$.

The termination condition was to quit the evolutionary loop if an improvement in the last $N = 10\,000$ generations was smaller than a predefined $\epsilon = 0.001$.

As the test-case generator has six parameters, it is difficult to fully discuss their interactions. Thus, we have selected a single point from the $\mathcal{TCG}(n, w, \lambda, \alpha, \beta, \mu)$ parameter search space:

$$n = 2;\ w = 10;\ \lambda = 1.0;\ \alpha = 0.9/\sqrt{n};$$
$$\beta = 0.1;\ \mu = 0.1$$

and have varied one parameter at a time. In each case, two figures summarize the results (which display averages of 100 runs).[9]

1) All left figures display the fitness value of the best feasible individual at the end of the run (continuous line), the average and the lowest heights among all feasible optima (broken lines).
2) All right figures rescale (linear scaling) the figure from the left: the fitness value of the best feasible individual at the end of the run (continuous line) and the average height among feasible peaks in the landscape (broken line) are displayed as fractions between 0 (which corresponds to the height of the lowest peak) and 1 (which corresponds to the height of the highest peak). Since the differences in peak heights among the best few peaks are not large, the scaled peak values give a better insight into the effect of each control parameter.

Fig. 10 displays the results of a static penalty method on the $\mathcal{TCG}$ while n is varied between 1 and 6. It is clear that an increase of n (dimensionality) reduces the efficiency of the algorithm: the value of the returned solution approaches the value of the average peak height in the landscape.

Fig. 11 displays the results of a static penalty method on the $\mathcal{TCG}$ while w is varied between 1 and 30 (for $w = 30$, the evaluation function has $w^n = 900$ peaks). An increase of w (multimodality) decreases the performance of the algorithm, but not to the extent we have seen in the previous case (increase of n). The reason is that, while $w = 10$ (Fig. 10), the number of peaks grows as 10^n (for $n = 3$, there are 1000 peaks), whereas for $n = 2$ (Fig. 11), the number of peaks grows as w^2 (for $w = 30$, there are 900 peaks only).

Fig. 12 displays the results of a static penalty method on the $\mathcal{TCG}$ while λ is varied between 0 and 1. It seems that the number of constraints of the test-case generator does not influence the performance of the system. However, it is important to underline again that the interpretation of constraints in the $\mathcal{TCG}$ is different from usual; see a discussion in Section II-A.

[9]In all cases, the values of standard deviations were similar and quite low.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

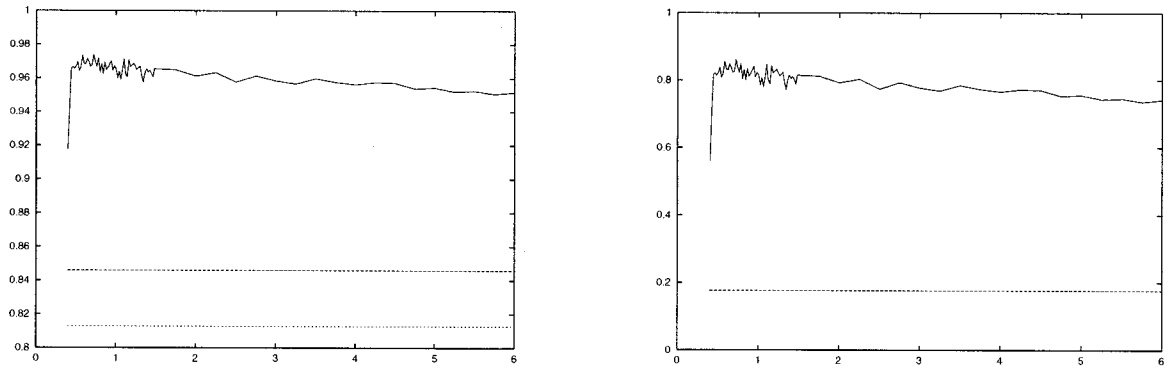

Fig. 17. Quality of solution found for different values of penalty coefficient W.

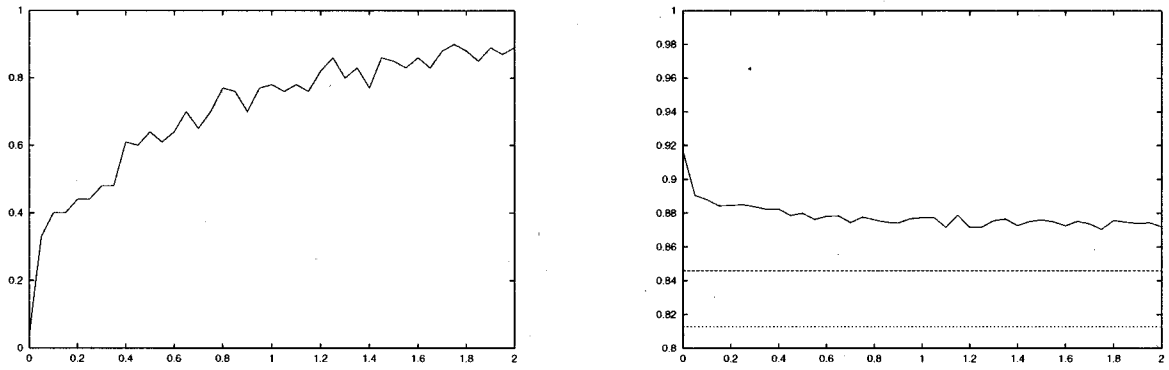

Fig. 18. Success rate (left graph) and quality of solution found (right graph) for different values of W, where a proportional selection replaced the tournament selection. Averages over 250 runs.

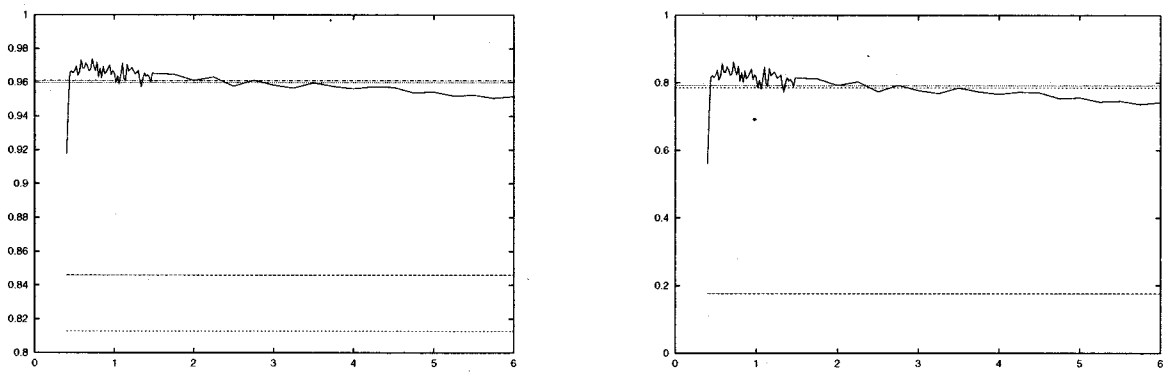

Fig. 19. Quality of solution found for two dynamic penalty methods versus previously discussed static penalty approach.

Fig. 13 displays the results of a static penalty method on the $\mathcal{TCG}$ while α is varied between 0 and $1/\sqrt{2}$. Clearly, larger values of α (better connectedness) improve the results of the algorithm, as it is easier to locate a feasible solution. Note an anomaly in the graph for $\alpha = 0$; this is due to (28), which provides for a different formula for a_k's when $\alpha\beta = 0$.

Fig. 14 displays the results of a static penalty method on the $\mathcal{TCG}$ while β is varied between 0 and 1. Larger values of β (smaller feasibility) decrease the size of rings; however, this feature does not seem to influence the performance of the algorithm significantly.

Fig. 15 displays the results of a static penalty method on the $\mathcal{TCG}$ while μ is varied between 0 and 1. Higher values of μ (higher ruggedness) usually make the test case slightly harder as the heights of the peaks are changed: local maxima are of almost the same height as the global one. We expected to see a larger influence of the parameter μ on the results; however, as the subclass of functions under investigation is relatively easy, this was not the case.

We therefore conclude that the results obtained by an evolutionary algorithm using a static penalty approach depend mainly on the dimensionality n, the multimodality w, and the connectedness α. On the other hand, they do not depend significantly on the parameters λ, β, and μ.

Another series of experiments aimed at exploring the relationship between the value of the penalty coefficient W and the

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

results of the runs. Again, we have selected a single point from the parameter search space of $\mathcal{TCG}(n, w, \lambda, \alpha, \beta, \mu)$:

$$n = 5;\ w = 10;\ \lambda = 0.2;\ \alpha = 0.9/\sqrt{5};$$
$$\beta = 0.5;\ \mu = 0.1$$

and varied penalty coefficient W from 0 to 10^4. In the first experiment, we measured the ratio of feasible solutions returned by the system for different values of W; Fig. 16 displays the results for $0 < W \leq 6$ (averages of 250 runs).

It is interesting to note that the experiment confirmed a simple intuition: the higher the penalty coefficient is, the better the chance that the algorithm returns a feasible solution. For larger values of W (from 20 up to 10^4), the success rate was equal to 1 (which is the reason we did not show it in Fig. 16). As infeasible solutions usually are of no interest, we conclude that high penalty coefficients guarantee that the final solution returned by the system is feasible.

However, a separate issue (apart from feasibility) is the quality of the returned solution. The next two graphs (Fig. 17) clearly make the point. The left graph displays the average fitness value of the best feasible individual at the end of the run (continuous line), together with the average height and the lowest height among all feasible optima (broken lines). The right graph, on the other hand, rescales the one from the left: the fitness value of the best feasible individual at the end of the run (continuous line) and the average height among feasible peaks in the landscape (broken line) are displayed as fractions between 0 (which corresponds to the height of the lowest peak) and 1 (which corresponds to the height of the highest peak).

The (similar) graphs of Fig. 17 confirm another intuition connected with static penalties: it is difficult to guess the "correct" value of the penalty coefficient as different landscapes require different values of W**!** Note that low values of W (i.e., values below 0.4) produce poor quality results; on the other hand, for larger values of W (i.e., values larger than 1.5), the quality of the solutions drops slowly (it stabilizes later—for large W—at the level of 0.95). Thus, the best values of penalty coefficient W for the particular landscape of the $\mathcal{TCG}(5, 10, 0.2, 0.9/\sqrt{5}, 0.5, 0.1)$ (which has $n^w = 10^5$ local optima) are in the range [0.6, 0.75].

The results of experiments reported in Figs. 16 and 17 may trigger many additional questions. For example, 1) What is the role of the selection method in these experiments?, or 2) What are the merits of a dynamic penalty method as opposed to static approach? In the following, we address these two issues in turn.

In evaluating various constraint-handling techniques, it is important to take into account other components of the evolutionary algorithm. It is difficult to compare two constraint-handling techniques if they are incorporated in two algorithms with different selection methods, operators, parameters, etc. To illustrate the point, we have replaced the tournament selection by a proportional selection, leaving everything else in the algorithm without any change. The graphs of Fig. 18 display the results: the success rate (in terms of finding a feasible solution) and the quality of solution found.

As expected (Fig. 18, right), the performance of the algorithm drops in a significant way. Proportional selection is much more sensitive to the actual values of the evaluation function.

We have also experimented with two dynamic penalty approaches. In both cases, the maximum number of generations was set to $T = 20\,000$, and

$$W_1(t) = 10 \cdot \frac{2t}{T} \quad \text{and} \quad W_2(t) = 10 \cdot \left(\frac{2t}{T}\right)^2$$

where t is the current generation number. Thus, W_1 varies between 0 and 20 (linearly), whereas W_2 varies between 0 and 40 following a quadratic curve.

The results of these experiments are displayed in Fig. 19. As before, the right graph rescales the left (as the value of 0 corresponds to the value of the lowest peak in the landscape). The continuous line (on both graphs) indicates the quality of a feasible solution found (average of 250 runs) for static values of W (repetition of graphs of Fig. 17). The success rate for finding a feasible solution was 1 for both schemes. Two horizontal broken lines indicate the performance of the system for the two dynamic penalty approaches (W_2 being slightly better than W_1). The conclusions are clear: for most values of W (for all $W > 3$), both dynamic penalty schemes perform better than a static approach. However, it is possible to find a static value (in our case, say, $0.6 \leq W \leq 1.5$), where the static approach outperforms both dynamic schemes.

V. Summary

We have discussed briefly how the proposed test-case generator $\mathcal{TCG}(n, w, \lambda, \alpha, \beta, \mu)$ can be used for the evaluation of a constraint-handling technique. As explained in Section II-E, the parameter n controls the dimensionality of the test function, the parameter w controls the modality of the function, the parameter λ controls the number of constraints in the search space, the parameter α controls the connectedness of the feasible search space, the parameter β controls the ratio of the feasible to total search space, and the parameter μ controls the ruggedness of the test function.

We believe that such a constrained test problem generator should serve the purpose of testing any method for constrained parameter optimization. Moreover, one can also use the $\mathcal{TCG}$ for testing any method for unconstrained optimization (e.g., operators, selection methods, etc.). In the previous section, we have indicated how it can be used to evaluate the merits and drawbacks of one particular constraint-handling method (static penalties). Note that it is possible to analyze the performance of a method further by varying two or more parameters of the $\mathcal{TCG}$.

The proposed test-case generator is far from perfect. It defines a landscape which is a collection of sitewise optimizable functions, each defined on different subspaces of equal sizes. Because of that, all basins of attractions have the same size; moreover, all points at the boundary between two basins of attraction have the same value. The local optima are located in the centers of the hypercubes; all feasible regions are centered around the

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

local optima. Note also that, while we can change the number of constraints, there is precisely one (for $\alpha\beta > 0$) active constraint at the global optimum.

Some of these weaknesses can be corrected easily. For example, in order to avoid the the symmetry and equal-sized basin of attraction of all subspaces, we may modify the $\mathcal{TCG}$ in the following two ways. First, the parameter vector $\vec{x}$ may be transformed into another parameter vector $\vec{y}$ using a nonlinear mapping $\vec{y} = g(\vec{x})$. The mapping function may be chosen in such a way so as to have the lower and upper bounds of each parameter y_i equal to 0 and 1, respectively. Such a nonlinear mapping will make all subspaces of different size. Second, in order to make the feasible search space asymmetric, the center of each subspace for the outer hypersphere can be made different from that of the inner hypersphere. The following update of the center for the outer hypersphere can be used:

$$p_i^k = (l_i^k + u_i^k)/2 + \gamma_i^k \epsilon_i^k$$

where ϵ_i^k is a small number denoting the maximum difference allowed between the centers of inner and outer hyperspheres, and γ_i^k is a random number between 0 and 1. To avoid using another control parameter, ϵ_i^k can be assumed to be the same for all subspaces. This modification makes the feasible search space asymmetric, although the maximum at each subspace remains the same as in the original function.

It might be worthwhile to further modify the $\mathcal{TCG}$ to parameterize the number of active constraints at the optima. It seems necessary to introduce the possibility of a more gradual increment of the number of peaks. In the current version of the test case generator, $w = 1$ implies one peak, and $w = 2$ implies 2^n peaks (this was the reason for using low values for parameter n). Also, the difference between the lowest and highest values of the peaks in the search space are, in the present model, too small.

All of these limitations of the $\mathcal{TCG}$ diminish its significance for being a useful tool for modeling real-world problems, which may have quite different characteristics. Note also that it is necessary to develop an additional tool, which would map a real-world problem into a particular configuration of the test-case generator. In such a case, the test-case generator should be able to "recommend" the most suitable constraint-handling method.

However, even in its current form, the proposed $\mathcal{TCG}$ is a useful tool for analyzing any constraint-handling method in nonlinear programming (for any algorithm, not necessarily an evolutionary algorithm). The Web page http://www.daimi.au.dk/~marsch/TCG.html contains two files: the $\mathcal{TCG}$ file TCG.c (standard C), and TestTCG.c, an example file that shows how to use the $\mathcal{TCG}$.

REFERENCES

[1] T. Bäck, F. Hoffmeister, and H.-P. Schwefel, "A survey of evolution strategies," in *Proc. 4th Int. Conf. Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 2–9.

[2] T. Bäck and M. Schütz, "Intelligent mutation rate control in canonical genetic algorithms," in *Proc. 9th Int. Symp. Methodologies of Intell. Syst.*, Z. W. Ras and M. Michalewicz, Eds. Berlin, Germany: Springer-Verlag, 1996.

[3] J. C. Bean and A. B. Hadj-Alouane, "A dual genetic algorithm for bounded integer programs," Dept. Ind. Oper. Eng., Univ. Michigan, Tech. Rep. TR 92-53, 1992.

[4] G. Bilchev and I. Parmee, "Ant colony search vs. genetic algorithms," Plymouth Eng. Design Centre, Univ. Plymouth, Tech. Rep., 1995.

[5] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proc. 1st Eur. Conf. Artif. Life, Paris*. Cambridge, MA: M.I.T. Press/Bradford Book, 1991.

[6] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 61–69.

[7] ——, private communication, 1997.

[8] K. Deb, *Optimization for Engineering Design: Algorithms and Examples*. New Delhi, India: Prentice-Hall, 1995.

[9] ——, An efficient constraint handling method for genetic algorithms, in *Comput. Methods Appl. Mech. Eng.*, to be published.

[10] K. De Jong, "The analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, 1975. *Dissertation Abstr. Int.*, vol. 36, no. 10, pp. 5140B. (University Microfilms 76-9381).

[11] A. Eiben, P.-E. Raue, and Z. Ruttkay, "Genetic algorithms with multi-parent recombination," in *Proc. 3rd Conf. Parallel Problem Solving from Nature*. Berlin, Germany: Springer-Verlag, 1994, pp. 78–87. No. 866 in LNCS.

[12] A. Eiben and Z. Ruttkay, "Self-adaptivity for constraint satisfaction: Learning penalty functions," in *Proc. 3rd IEEE Conf. Evol. Comput.*. New York: IEEE, 1996, pp. 258–261.

[13] L. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundations of Genetic Algorithms 2*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 187–202.

[14] C. Floudas and P. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*. Berlin, Germany: Springer-Verlag, 1987, vol. 455, LNCS.

[15] J. Gregory. (1995) Nonlinear programming FAQ. Usenet sci.answers. [Online]. Available: ftp://rtfm.mit.edu/pub/usenet/sci.answers/nonlinear-programming-faq

[16] A. B. Hadj-Alouane and J. C. Bean, "A genetic algorithm for the multiple-choice integer program," Dept. Ind. Oper. Eng., Univ. Michigan, Tech. Rep. TR 92-50, 1992.

[17] D. Himmelblau, *Applied Nonlinear Programming*. New York: McGraw-Hill, 1992.

[18] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*. Berlin, Germany: Springer-Verlag, 1981, vol. 187, Lecture Notes in Econ. and Math. Syst..

[19] A. Homaifar, S. H.-Y. Lai, and X. Qi, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–254, 1994.

[20] J. Joines and C. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with gas," in *Proc. 1st IEEE Int. Conf. Evol. Comput.*, Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, Eds. Piscataway, NJ: IEEE, 1994, pp. 579–584.

[21] A. J. Keane, "A brief comparison of some evolutionary optimization methods," in *Modern Heuristic Search Methods*, V. Rayward-Smith, I. Osman, C. Reeves, and G. D. Smith, Eds. New York: Wiley, 1996, pp. 255–272.

[22] J. Kelly and M. Laguna. (1996, Apr.) Article posted in *Genetics Algorthm Dig*. [Online]

[23] S. Kozieł, "Evolutionary algorithms in constrained numerical optimization problems on convex spaces," *Electron. Telecommun. Quart.*, vol. 43, no. 1, pp. 5–18, 1997.

[24] S. Kozieł and Z. Michalewicz, "A decoder-based evolutionary algorithm for constrained parameter optimization problems," in *Proc. 5th Conf. Parallel Problems Solving from Nature*. Berlin, Germany: Springer-Verlag, 1998.

[25] ——, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evol. Comput.*, 1999.

[26] R. G. Leriche, C. Knopf-Lenoir, and R. T. Haftka, "A segragated genetic algorithm for constrained structural optimization," in *Proc. 6th Int. Conf. Genetic Algorithms*, L. J. Eshelman, Ed., 1995, pp. 558–565.

[27] C. Maa and M. Shanblatt, "A two-phase optimization neural network," *IEEE Trans. Neural Networks*, vol. 3, pp. 1003–1009, June 1992.

[28] Z. Michalewicz, "Genetic algorithms, numerical optimization and constraints," in *Proc. 6th Int. Conf. Genetic Algorithms*, L. J. Eshelman, Ed. San Mateo, CA: Morgan Kaufmann, 1995, pp. 151–158.

[29] ——, "Heuristic methods for evolutionary computation techniques," *J. Heuristics*, vol. 1, no. 2, pp. 177–206, 1995b.

[30] ——, "A survey of constraint handling techniques in evolutionary computation methods," in *Proc. 4th Annu. Conf. Evolutionary Programming*. Cambridge, MA: M.I.T. Press, 1995.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

[31] ——, *Genetic Algorithms+Data Structures=Evolution Programs*, 3rd ed. New York: Springer-Verlag, 1996.
[32] Z. Michalewicz and N. Attia, "Evolutionary optimization of constrained problems," in *Proc. 3rd Annu. Conf. Evol. Programming*, A. V. Sebald and L. J. Fogel, Eds. Singapore: World Scientific, 1994, pp. 98–108.
[33] Z. Michalewicz, D. Dasgupta, R. G. Le Riche, and M. Schoenauer, "Evolutionary algorithms for constrained engineering problems," *Comput. Ind. Eng. J.*, vol. 30, no. 2, pp. 851–870, 1996.
[34] Z. Michalewicz, K. Deb, M. Schmidt, and T. Stidsen, "Evolutionary algorithms for engineering applications," in *Proc. EUROGEN'99*, Jyväskylä, Finland, May 1999, pp. 73–94.
[35] ——, "Towards understanding constraint-handling methods in evolutionary algorithms," in *Proc. 1999 Congr. Evolutionary Computation*, Washington, DC, July 1999, pp. 1459–1464.
[36] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. Berlin, Germany: Springer-Verlag, 2000.
[37] Z. Michalewicz and C. Z. Janikow, "Handling constraints in genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, R. K. Belew and L. B. Booker, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 151–157.
[38] Z. Michalewicz, T. Logan, and S. Swaminathan, "Evolutionary operators for continuous convex parameter spaces," in *Proc. 3rd Annu. Conf. Evolutionary Programming*, A. V. Sebald and L. J. Fogel, Eds. Singapore: World Scientific, 1994, pp. 84–97.
[39] Z. Michalewicz and M. Michalewicz, "Pro-life versus pro-choice strategies in evolutionary computation techniques," in *Evolutionary Comput.*. Piscataway, NJ: IEEE, 1995, ch. 10.
[40] Z. Michalewicz and G. Nazhiyath, "Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints," in *Proc. 2nd IEEE Int. Conf. Evolutionary Computation*, D. B. Fogel, Ed. Piscataway, NJ: IEEE, 1995, pp. 647–651.
[41] Z. Michalewicz, G. Nazhiyath, and M. Michalewicz, "A note on usefulness of geometrical crossover for numerical optimization problems," in *Proc. 5th Annu. Conf. Evol. Programming*, P. J. Angeline, L. J. Fogel, and T. Bäck, Eds. Cambridge, MA, 1996.
[42] Z. Michalewicz and M. Schoenauer, "Evolutionary computation for constrained parameter optimization problems," *Evol. Comput.*, vol. 4, no. 1, pp. 1–32, 1996.
[43] H. Mühlenbein and H.-M. Voigt, "Gene pool recombination for the breeder genetic algorithm," in *Proc. Metaheuristics Int. Conf.*, 1995, pp. 19–25.
[44] H. Myung, J.-H. Kim, and D. Fogel, "Preliminary investigation into a two-stage method of evolutionary optimization on constrained problems," in *Proc. 4th Annu. Conf. Evol. Programming*, J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, Eds. Cambridge, MA: M.I.T. Press, 1995, pp. 449–463.
[45] D. Orvosh and L. Davis, "Shall we repair? Genetic algorithms, combinatorial optimization, and feasibility constraints," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, p. 650.
[46] C. C. Palmer and A. Kershenbaum, "Representing trees in genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, June 1994, pp. 379–384.
[47] J. Paredis, "Coevolutionary constraint satisfaction," in *Proc. 3rd Conf. Parallel Problem Solving from Nature*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Berlin, Germany: Springer-Verlag, 1994, pp. 46–55.
[48] I. Parmee and G. Purchase, "The development of directed genetic search technique for heavily constrained design spaces," in *Proc. Conf. Adaptive Comput. Eng. Design Contr.*, 1994, pp. 97–102. Univ. Plymouth.
[49] D. Powell and M. M. Skolnick, "Using genetic algorithms in engineering design optimization with non-linear constraints," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 424–430.
[50] I. Rechenberg, *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Stuttgart, Germany: Fromman-Holzboog Verlag, 1973.
[51] J.-M. Renders and H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways," in *Proc. 1st IEEE Int. Conf. Evol. Comput.*, Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, Eds. Piscataway, NJ: IEEE, 1994, pp. 312–317.
[52] R. Reynolds, "An introduction to cultural algorithms," in *Proc. 3rd Annu. Conf. Evol. Programming*, A. V. Sebald and L. J. Fogel, Eds. Singapore: World Scientific, 1994, pp. 131–139.
[53] R. Reynolds, Z. Michalewicz, and M. Cavaretta, "Using cultural algorithms for constraint handling in Genocop," in *Proc. 4th Annu. Conf. Evol. Programming*, J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, Eds. Cambridge, MA: M.I.T. Press, 1995, pp. 298–305.
[54] J. T. Richardson, M. R. Palmer, G. Liepins, and M. Hilliard, "Some guidelines for genetic algorithms with penalty functions," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 191–197.
[55] G. Rudolph, *Convergence Properties of Evolutionary Algorithms*. Hamburg, Germany: Verlag Dr. Kovacheckc, 1996.
[56] D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, J. J. Grefenstette, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1985.
[57] M. Schoenauer and Z. Michalewicz, "Evolutionary computation at the edge of feasibility," in *Proc. 4th Conf. Parallel Problem Solving from Nature*, I Rechenberg, H.-P. Schuefel, W. Ebeling, and H.-M. Voigt, Eds. Berlin, Germany: Springer-Verlag, 1996.
[58] ——, "Boundary operators for constrained parameter optimization problems," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Bäck, Ed. San Mateo, CA: Morgan Kaufmann, 1997, pp. 320–329.
[59] M. Schoenauer and S. Xanthakis, "Constrained GA optimization," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 573–580.
[60] H.-P. Schwefel, *Numerical Optimization of Computer Models*, 2nd ed. New York: Wiley, 1981, 1995.
[61] A. Smith and D. Tate, "Genetic optimization using a penalty function," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 499–503.
[62] P. Surry, N. Radcliffe, and I. Boyd, "A multi-objective approach to constrained optimization of gas supply networks," in *Proc. AISB-95 Workshop Evol. Comput.*, T. Fogarty, Ed. Berlin, Germany: Springer-Verlag, 1995, vol. 993, pp. 166–180.
[63] C. H. M. van Kemenade, "Recombinative evolutionary search," Ph.D. dissertation, Leiden Univ., Leiden, The Netherlands, 1998.
[64] D. Waagen, P. Diercks, and J. McDonnell, "The stochastic direction set algorithm: A hybrid technique for finding function extrema," in *Proc. 1st Annu. Conf. Evol. Programming*, D. B. Fogel and W. Atmar, Eds: Evol. Programming Soc., 1992, pp. 35–42.
[65] D. Whitley, K. Mathias, S. Rana, and J. Dzubera, "Building better test functions," in *Proc. 6th Int. Conf. Genetic Algorithms*, L. Eshelman, Ed. San Mateo, CA: Morgam Kaufmann, 1995.
[66] ——, "Evaluating evolutionary algorithms," *Artif. Intell. J.*, vol. 85, pp. 245–276, Aug. 1996.
[67] M. Wodrich and G. Bilchev, "Cooperative distributed search: The ant's way," *Contr. Cybern.*, vol. 26, no. 3, 1997.
[68] A. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms*, J. G. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 205–218.

Zbigniew Michalewicz received the M.Sc. degree from the Technical University of Warsaw in 1974, and the Ph.D. degree from the Institute of Computer Science, Polish Academy of Sciences in 1981.

He is a Professor of Computer Science at the University of North Carolina at Charlotte. His current research interests are in the field of evolutionary computation. Dr. Michalewicz has published several books, including a monograph (three editions), and over 160 technical papers in journals and conference proceedings. He was the General Chairman of the 1st IEEE International Conference on Evolutionary Computation, Orlando, FL, June 1994. He is also Chief Scientist with Nutech Solutions Inc., Charlotte, NC. He has been an invited speaker at many international conferences, and a member of 40 various program committees of international conferences during the last three years. He is a current member of the Editorial Board and/or serves as an Associate Editor for nine international journals (including IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION). Recently, he published (together with D. B. Fogel) *How to Solve It: Modern Heuristics* (Berlin, Germany: Springer-Verlag, 2000).

Kalyanmoy Deb is a Professor in the Mechanical Engineering Department, Indian Institute of Technology, Kanpur. His research interest is in the area of developing and applying optimization algorithms, particularly using evolutionary computation. He is the author of an optimization textbook (Prentice-Hall of India). He has over 90 publications, and is actively involved in promoting the research and practice of evolutionary algorithms in India and abroad through his Kanpur Genetic Algorithms Laboratory (KanGAL). He is also an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION

Dr. Deb is one of the 15 Executive Council members of the newly formed International Society on Genetic and Evolutionary Computation (ISGEC).

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.

Martin Schmidt received the M.Sc. degree in computer science from the University of Aarhus in 1997.

He is a Ph.D. degree student at the Institute of Computer Science, Polish Academy of Sciences, with Prof. Z. Michalewicz as his supervisor. His current research interests are in the field of evolutionary computation. He is also actively involved in the application of evolutionary algorithms in business. He has published over ten papers in international conference proceedings and journals, and has also published chapters in John Wiley and Springer-Verlag books. Currently, he is a Senior Manager with the Computational Intelligence Division, Nutech Solutions Inc., Charlotte, NC.

Thomas Stidsen received the M.Sc. degree in computer science from the University of Aarhus in 1997.

He is currently a Ph.D. degree student in the Operations Research Section, Technical University of Denmark, with Prof. J. Clausen as his supervisor. The Ph.D. project deals with the design and optimization of optical telecommunication networks, using both heuristic optimization methods and optimal optimization methods. He has published five papers in international conference proceedings and in journals.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY KANPUR. Downloaded on March 16,2010 at 05:36:46 EDT from IEEE Xplore. Restrictions apply.