Available online at www.sciencedirect.com

SCIENCE DIRECT®

Progress in Nuclear Energy 48 (2006) 525–539

PROGRESS IN NUCLEAR ENERGY
An International Review Journal

www.elsevier.com/locate/pnucene

Two stochastic optimization algorithms applied to nuclear reactor core design

Wagner F. Sacco [a,b,*], Cassiano R.E. de oliveira [a], Cláudio M.N.A. Pereira [b,c]

[a] *Nuclear and Radiological Engineering Program, George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0405, USA*

[b] *Comissão Nacional de Energia Nuclear, DIRE/IEN, Ilha do Fundão s/n, 21945-970, PO Box 68550, Rio de Janeiro, Brazil*

[c] *Universidade Federal do Rio de Janeiro – PEN/COPPE, Ilha do Fundão s/n, 21945-970, PO Box 68509, Rio de Janeiro, Brazil*

Received 9 May 2005; revised 30 October 2005; accepted 30 October 2005

Abstract

Two stochastic optimization algorithms conceptually similar to Simulated Annealing are presented and applied to a core design optimization problem previously solved with Genetic Algorithms. The two algorithms are the novel Particle Collision Algorithm (PCA), which is introduced in detail, and Dueck's Great Deluge Algorithm (GDA). The optimization problem consists in adjusting several reactor cell parameters, such as dimensions, enrichment and materials, in order to minimize the average peak factor in a three-enrichment-zone reactor, considering restrictions on the average thermal flux, criticality and sub-moderation. Results show that the PCA and the GDA perform very well compared to the canonical Genetic Algorithm and its variants, and also to Simulated Annealing, hence demonstrating their potential for other optimization applications.

Keywords: Metaheuristics; Stochastic optimization; Nuclear reactor design

1. Introduction

Stochastic optimization methods based on the Simulated Annealing paradigm (Kirkpatrick et al., 1983) have been actively developed in the last 30 years and successfully applied to numerous complex optimization problems in engineering and medical sciences. These methods though very powerful are not free from practical drawbacks, the main one being that performance is too sensitive to the choice of free parameters, such as, for example, the annealing schedule and initial temperature (Carter, 1997).

Ideally, an optimization algorithm should not rely on user-defined or problem-dependent parameters and should not converge to a suboptimal solution. Given the appropriate annealing schedule, Simulated Annealing is guaranteed to converge to the global optimum (Aarts and Korst, 1989), as it is a Metropolis-based algorithm (Metropolis et al., 1953), where a worse solution can be accepted with a certain probability. However, its rate of convergence is strongly

* Corresponding author. Nuclear and Radiological Engineering Program, The George Woodruff School of Mechanical Engineering, 900 Atlantic Drive NW, Neely Building, Room G108, Atlanta, GA 30332-0405, USA. Tel.: +1 404 385 6890; fax: +1 404 894 3733.
E-mail address: wagner.sacco@me.gatech.edu (W.F. Sacco).

0149-1970/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.pnucene.2005.10.004

dependent on the user-specified initial parameters. A desirable outcome would be, therefore, the development of an algorithm similar to Simulated Annealing, but without the burden of parameter specification.

One such algorithm, which does not rely on user-supplied parameters to perform the optimality search, is introduced in detail here. Its structure was outlined, along with some preliminary results, in Sacco and Oliveira (2005). Named the "Particle Collision Algorithm" (PCA), the algorithm is loosely inspired by the physics of nuclear particle collision reactions (Duderstadt and Hamilton, 1976), particularly scattering and absorption. Thus, a particle that hits a high-fitness "nucleus" would be "absorbed" and would explore the boundaries. On the other hand, a particle that hits a low-fitness region would be scattered to another region. This permits us to simulate the exploration of the search space and the exploitation of the most promising areas of the fitness landscape through successive scattering and absorption collision events.

Another algorithm conceptually similar to Simulated Annealing that is presented here is the Great Deluge Algorithm (GDA) (Dueck, 1993). It is an analogy with a flood: the "water level" rises continuously and the proposed solution must lie above the "surface" in order to survive. The user must specify two parameters: the "rain speed", which controls convergence of the algorithm similar to SA's annealing schedule, and the initial rain level, analogous to SA's initial temperature. The main advantage of this algorithm in relation to Simulated Annealing is its robustness to parameter specification (Bykov, 2003).

The remainder of the paper is organized as follows. The section following presents an overview of the established optimization methods, namely, Simulated Annealing, Great Deluge Algorithm and Genetic Algorithm, which are used in the comparative tests. Section 3 presents the details of the implementation of the new PCA method and its validation tests. Section 4 provides a description of the reactor design optimization problem, details of the numerical implementation of the algorithms, and the numerical comparisons. Finally, in Section 5, conclusions are presented along with suggestions for future improvements to PCA.

2. Overview of established optimization methods

2.1. Simulated Annealing

Simulated Annealing was introduced as a computational method mimicking the physical process of the increasing of energetic stability of molecular structure by consecutive heating and cooling of a material. The candidate solutions with worse objective function values are accepted with a probability given by

$$P = \exp(\Delta E/T), \tag{1}$$

where ΔE is the energy variation from two consecutive states and T is the current temperature. The temperature reduction scheme is known as the "cooling schedule". This can involve arbitrarily reducing the temperature after a certain number of iterations or successful moves, or by defining a simple progression formula:

$$T = T_0 \alpha^n \tag{2}$$

where T_0 is the initial temperature, α is a number between 0.9 and 0.99 and n is the iteration number (Aarts and Korst, 1989). Fig. 1 shows the SA's pseudo code for a maximization problem.

2.2. The Great Deluge Algorithm

As mentioned previously, the Great Deluge Algorithm draws an analogy with flood phenomena. Like Simulated Annealing, GDA may accept worse candidate solutions than the current best during its run. The worse solution is accepted if its fitness is higher than the water level, which is the control parameter.

The water level, WL, receives an initial value WL_0 which is increased iteratively by the "rain speed". We use here the rain speed, Up, suggested by Bykov (2003), which is given by:

$$\text{Up} = \frac{f(x') - \text{WL}_0}{N}, \tag{3}$$

where $f(x')$ is a goal value and N is the number of iterations. This goal value can be estimated by some quick technique (e.g. Hill-Climbing) or by some previously known results. The initial water level can be set as the lower boundary of the results. But as mentioned, this algorithm is relatively insensitive to this parameter.

```
Choose an initial configuration Old_Config
Choose an initial temperature $T_0$
For n = 0 to # of iterations
    Generate a small stochastic perturbation New_Config of the solution
    Compute $\Delta E$ = Fitness(New_Config) - Fitness(Old_Config)
    If $\Delta E > 0$
        Old_Config := New_Config
    Else
        With probability $\exp(\Delta E/T)$
            Old_Config := New_Config
    End If
    Reduce temperature
End For
```

Fig. 1. Pseudo code for SA.

Fig. 2 shows the GDA's pseudo code. The algorithm's default is for maximization problems. For minimization problems, fitness values must be multiplied by -1.

2.3. Genetic Algorithms

Genetic Algorithms (GAs) (Goldberg, 1989) are search methods based upon the biological principles of natural selection and survival of the fittest as introduced by Charles Darwin in his seminal work ‘‘The Origin of Species’’ (1859). They were rigorously introduced by Holland (1975). GAs consist of a population of individuals that are possible solutions and each one of these individuals receives a reward, known as ‘‘fitness’’, that quantifies its suitability to solve the problem. In ordinary applications, fitness is simply the objective function. Individuals with better than average fitnesses receive greater opportunities to cross. On the other hand, low-fitness individuals will have less chance to reproduce until they are extinguished. Consequently, the good features of the best individuals are disseminated over the generations. In other words, the most promising areas of the search space are explored, making the GA converge to the optimal or near-optimal solution.

Genetic Algorithms have proven to be efficient in a great variety of areas of application, as the population of candidate solutions converge to a single optimum, in a phenomenon known as genetic drift (Goldberg, 1989). Many populational diversity mechanisms, called niching methods (Mahfoud, 1995), have been proposed to force the GA to maintain a heterogeneous population throughout the evolutionary process. These methods are inspired by nature, as in an ecosystem there are different subsystems (niches) that contain many diverse species (subpopulations). The number of elements in a niche is determined by its resources and by the efficiency of each individual in taking profit of these resources. Each peak of the multimodal function can be seen as a niche that supports a number of individuals directly

```
Choose an initial configuration Old_Config
Choose $WL_0$ and Up
For n = 0 to # of iterations
    Generate a small stochastic perturbation New_Config of the solution
    If Fitness(New_Config) > WL
        Old_Config := New_Config
    End If
    $WL = WL + Up$
End For
```

Fig. 2. Pseudo code for GDA.

proportional to its ‘‘fertility’’, which is measured by the fitness of this peak relative to the fitnesses of the other peaks of the domain. Genetic Algorithms which employ this technique are known as ‘‘Niching Genetic Algorithms’’ (NGAs).

The niching method referred herein is Fuzzy Clearing (Sacco et al., 2004), where the individuals are clustered using an algorithm called ‘‘Fuzzy Clustering Means’’ (Bezdek, 1981) and the individual with best fitness (dominant) is determined for each cluster. Following that the dominant’s fitness is preserved and all the others’ individuals have their fitnesses zeroed (in the case of a maximization problem).

Another way of preserving populational diversity is by using the Island Genetic Algorithm (IGA) (Cantu-Paz, 2000). In this model, the population is divided into multiple populations that evolve isolated from each other most of the time, but exchange individuals occasionally with their neighbors (migration).

3. The Particle Collision Algorithm (PCA)

3.1. Algorithm description

The PCA resembles in its structure that of Simulated Annealing: first an initial configuration is chosen; then there is a modification of the old configuration into a new one. The qualities of the two configurations are compared. A decision then is made on whether the new configuration is ‘‘acceptable’’. If it is, it serves as the old configuration for the next step. If it is not acceptable, the algorithm proceeds with a new change of the old configuration. PCA can also be considered a Metropolis algorithm, as a trial solution can be accepted with a certain probability. This acceptance may avoid the convergence to local optima.

The pseudo code description of the PCA is shown in Fig. 3. The algorithm’s default is for maximization problems. For minimization, just multiply the objective function by -1 and invert the ratio in $p_{\text{scattering}}$.

```
Generate an initial solution Old_Config
Best Fitness = Fitness (Old_Config)
For n = 0 to # of iterations
    Perturbation()
    If Fitness(New_Config) > Fitness(Old_Config)
        If Fitness(New_Config) > Best Fitness
            Best Fitness := Fitness(New_Config)
        End If
        Old_Config := New_Config
        Exploration ()
    Else
        Scattering ()
    End If
End For

Exploration ()
    For n = 0 to # of iterations
        Small_Perturbation()
        If Fitness(New_Config) > Fitness(Old_Config)
            Old_Config := New_Config
        End If
    End For
return

Scattering ()
    $p_{scattering} = 1 - \frac{Fitness(New\_Config)}{Best\ Fitness}$
    If $p_{scattering}$ > random (0, 1)
        Old_Config := random solution
    Else
        Exploration ();
    End if
return
```

Fig. 3. Pseudo code for PCA.

```
Perturbation()
    For i = 0 to (Dimension-1)
        Upper := Superior Limit [i]
        Lower := Inferior Limit [i]
        Rand = Random(0,1)
        New_Config[i] := Old_Config[i] + ((Upper - Old_Config[i])*Rand) -
                         ((Old_Config[i] - Lower)*(1-Rand))
    If (New_Config[i] > Upper)

        New_Config[i] := SupLim[i];
    Else
        If (New_Config[i] < Lower)

            New_Config[i] := InfLim[i];
        End If
    End If
End
```

Fig. 4. Function ‘‘Perturbation’’.

The ‘‘stochastic perturbations’’ in the beginning of the loop are random variations in each variable’s values within their ranges. We based this mechanism on that used in the Simulated Annealing algorithm (Aarts and Korst, 1989). For details, see function ‘‘Perturbation’’ in Fig. 4.

If the quality or fitness of the new configuration is better than the fitness of the old configuration, then the ‘‘particle’’ is ‘‘absorbed’’, there is an exploration of the boundaries searching for an even better solution. Function ‘‘Exploration()’’ performs this local search, generating a small stochastic perturbation of the solution inside a loop. In PCA’s current version, it is a one-hundred-iteration loop. The ‘‘small stochastic perturbation’’ is similar to the previous stochastic perturbation, but in a smaller range (see ‘‘Small_Perturbation’’ in Fig. 5).

Otherwise, if the quality of the new configuration is worse than the old configuration, the ‘‘particle’’ is ‘‘scattered’’. By scattering we mean that the new configuration receives random values between the upper and lower bounds of each design variable. The scattering probability ($p_{\text{scattering}}$) is inversely proportional to its quality. A low-fitness particle will have a greater scattering probability.

3.2. Validation

As the search space of the reactor design optimization problem is not known beforehand, prior to applying PCA to this problem, it was deemed necessary to validate the algorithm. We applied commonly employed test functions and compared the results to those obtained by the more established genetic and Simulated Annealing algorithms, and also by the conceptually similar GDA. These functions, which are defined below, have a single global optimum. In case of multiple global optima (same values for the objective function at different locations), these algorithms would all converge to one of these optima. The only method that can handle this situation is the Niching Genetic Algorithm (Mahfoud, 1995).

3.3. Test functions

Easom’s (1990) function is a unimodal test function where the global minimum region has a small area relative to the search space (see Fig. 6, below), making it a great challenge for optimization algorithms.

$$z(x,y) = -\cos(x)\cos(y)\exp\left\{-1\left[(x-\pi)^2+(y-\pi)^2\right]\right\},\ -100 \le x,y \le 100, \tag{4}$$

with global minimum $z = -1$ at $(x,y) = (\pi,\pi)$.

```
Small_Perturbation()
    For i = 0 to (Dimension-1)
        Upper = Random(1.0, 1.2) * Old_Config[i]
        If (Upper > Superior Limit [i])
            Upper = Superior Limit [i]
        End If
        Lower = Random(0.8, 1.0) * Old_Config[i]
        If (Lower < Inferior Limit [i])
            Lower = Inferior Limit [i]
        End If
        Rand = Random(0,1)
        New_Config[i] = Old_Config[i] + ((Upper - Old_Config[i])*Rand) -
                    ((Old_Config[i] - Lower)*(1-Rand))
End
```

Fig. 5. Function “Small_Perturbation”.

Shekel’s Foxholes, introduced by Shekel (1971) and adapted for maximization by De Jong (1975), is a two-dimensional function with 25 peaks all with different heights, ranging from 476.191 to 499.002 (Fig. 7). The global optimum is located at (−32,−32). Shekel’s Foxholes is defined by:

$$z(x,y) = 500 - \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{1 + i + (x - a(i))^6 + (y - b(i))^6}}, -65.536 \le x, y \le 65.536, \tag{5}$$

where $a(i) = 16[(i \bmod 5) - 2]$ and $b(i) = 16[(i/5) - 2]$.

Due to the high modality of Shekel’s Foxholes function it is a difficult task to determine its points of local and global maxima.

The last test function, Rosenbrock’s valley, is a classic optimization problem (Rosenbrock, 1960). The global optimum is inside a long, narrow, parabolic shaped flat valley (see Fig. 8). To find the valley is trivial, however, convergence to the global optimum is difficult and hence this problem has been repeatedly used to test the performance of optimization algorithms. The global optimum of the function, defined by Eq. (6), is $z = 0$ at $(x,y) = (1,1)$.

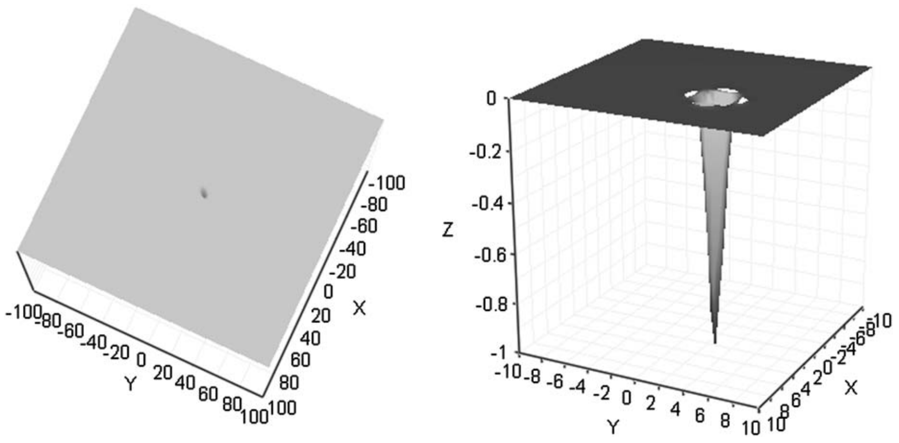

Fig. 6. Easom’s function.

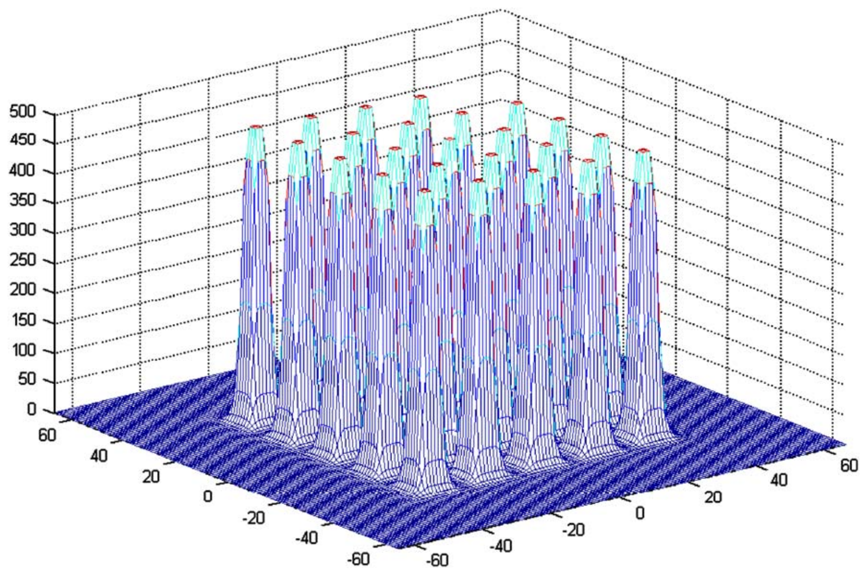

Fig. 7. Shekel's Foxholes.

$$z(x,y) = 100\left(y-x^2\right)^2 + (1-x)^2,\ -2.048 \le x,y \le 2.048 \tag{6}$$

3.4. Results

The Standard Genetic Algorithm (SGA) used in the validation tests was that implemented in GENESIS (Greffenstette, 1990), which employs double-point crossover (Goldberg, 1989), Stochastic Universal Sampling as the selection scheme (Baker, 1987) and elitism (Goldberg, 1989). The following parameters were used, as recommended by Goldberg (1989): 100,000 function evaluations (population size = 100, number of generations = 1000), crossover rate = 0.6, mutation rate = 0.005, generation gap = 0.1. The SA, PCA and GDA were set up for 100,000 function evaluations, the cooling schedule used in the SA was given by Eq. (2) with $\alpha = 0.95$ and the rain speed in the GDA followed Eq. (3). Various initial temperatures and initial rain levels were tested for SA and GDA so that the most suitable values were used for each function. The relatively high number of function evaluations was used to allow the

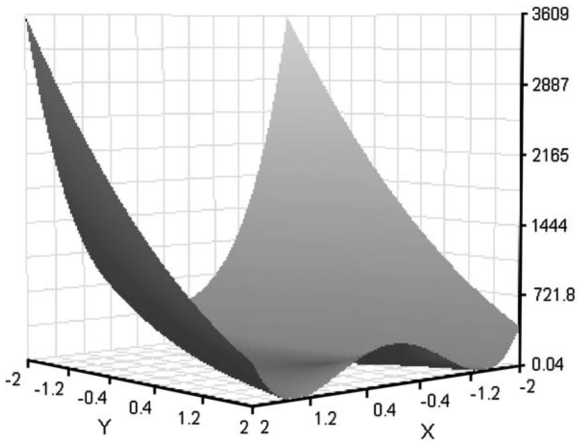

Fig. 8. Rosenbrock's valley.

Table 1
Results for Easom's function

Experiment	SGA		SA		GDA		PCA	
	Best	Iter.	Best	Iter.	Best	Iter.	Best	Iter.
Best results								
#1	−1.000	800	−1.000	800	−1.000	400	−1.000	200
#2	−1.000	3100	−1.000	1100	−1.000	900	−1.000	200
#3	−1.000	3800	−1.000	1200	−1.000	2100	−1.000	200
#4	−1.000	3900	−1.000	1200	−1.000	2400	−1.000	200
#5	−1.000	4100	−1.000	1800	−1.000	2500	−1.000	400
#6	−1.000	4200	−1.000	3400	−1.000	2900	−1.000	400
#7	−1.000	4400	−1.000	3900	−1.000	3100	−1.000	400
#8	−1.000	4700	−1.000	4400	−1.000	3200	−1.000	500
#9	−1.000	4800	−1.000	4800	−1.000	3400	−1.000	500
#10	−1.000	5100	−1.000	5200	−1.000	3800	−1.000	500
Average	−1.000	3890	−1.000	2780	−1.000	2470	−1.000	350
Std. Dev.	0.000	1266	0.000	1729	0.000	1087	0.000	135
Worst results								
#1	−1.000	40,100	−1.000	36,900	−1.000	32,600	−1.000	10,000
#2	−1.000	43,400	−1.000	37,300	−1.000	32,800	−1.000	10,000
#3	0.000	1700	−1.000	37,400	−1.000	33,000	−1.000	10,500
#4	0.000	2100	−1.000	37,800	−1.000	33,200	−1.000	11,600
#5	0.000	2400	−1.000	39,400	−1.000	33,400	−1.000	12,000
#6	0.000	2400	−1.000	39,900	−1.000	33,500	−1.000	12,400
#7	0.000	2900	−1.000	40,000	−1.000	33,800	−1.000	13,000
#8	0.000	3100	−1.000	40,300	−1.000	34,700	−1.000	13,400
#9	0.000	3900	−1.000	40,500	−1.000	35,000	−1.000	17,400
#10	0.000	5900	−1.000	40,900	−1.000	35,100	−1.000	20,600
Average	−0.200	10,790	−1.000	39,040	−1.000	33,710	−1.000	13,090
Std. Dev.	0.422	16,378	0.000	1520	0.000	916	0.000	3412
All executions								
Average	−0.920	11,257	−1.000	21,234	−1.000	18,845	−1.000	3550
Std. Dev.	0.273	8058	0.000	11,504	0.000	0.000	0.000	3804

algorithms to converge. As the GDA and PCA are by default maximization algorithms, it was necessary to multiply Easom's function and Rosenbrock's valley equations by −1, as it would be for any minimization problem.

Tables 1–3 display the results obtained by each algorithm. We performed 100 independent runs with different random seeds for each algorithm. The tables below present the 10 best and 10 worst values obtained in each run and the number of evaluations required to reach them, and also the averages and standard deviations for the 100 executions.

The results show that PCA reached the optimal values in all tests, requiring less function evaluations than the other methods. All function evaluations were counted, including those in PCA's "Exploration" phase. In the case of Easom's function, because of the extremely narrow valley, the Standard Genetic Algorithm drifted in some executions before reaching this region.

4. Numerical comparisons

4.1. Problem description

The main objective of this work was to compare two SA-like algorithms with Simulated Annealing (SA) and Genetic Algorithms (SGA and the NGA), and for this purpose we use the nuclear reactor design problem previously described in Pereira et al. (1999). Briefly this problem consists of a cylindrical 3-enrichment-zone PWR, with a typical cell composed by moderator (light water), cladding and fuel, see Fig. 9.

The design parameters that may be varied in the optimization process, as well as their variation ranges, are shown in Table 4. The materials are represented by discrete variables.

Table 2
Results for Shekel's Foxholes

Experiment	SGA		SA		GDA		PCA	
	Best	Iter.	Best	Iter.	Best	Iter.	Best	Iter.
Best results								
#1	499.002	200	499.002	200	499.002	100	499.002	100
#2	499.002	500	499.002	400	499.002	200	499.002	100
#3	499.002	600	499.002	600	499.002	300	499.002	100
#4	499.002	800	499.002	600	499.002	500	499.002	100
#5	499.002	900	499.002	800	499.002	600	499.002	100
#6	499.002	1000	499.002	1100	499.002	700	499.002	100
#7	499.002	1000	499.002	1100	499.002	700	499.002	200
#8	499.002	1100	499.002	1300	499.002	1100	499.002	200
#9	499.002	1100	499.002	1300	499.002	1600	499.002	200
#10	499.002	1400	499.002	1400	499.002	1900	499.002	300
Average	499.002	860	499.002	880	499.002	770	499.002	150
Std. Dev.	0.000	347	0.000	418	0.000	595	0.000	71
Worst results								
#1	498.795	17,700	499.002	36,900	499.002	31,000	499.002	15,700
#2	498.795	19,600	499.002	43,700	499.002	32,500	499.002	16,900
#3	498.795	24,000	499.002	45,100	499.002	33,000	499.002	17,600
#4	498.795	24,700	499.002	46,000	499.002	33,500	499.002	22,500
#5	498.795	30,700	499.002	50,400	499.002	33,600	499.002	23,600
#6	498.795	50,500	499.002	51,500	499.002	34,900	499.002	25,000
#7	498.588	17,500	499.002	59,200	499.002	36,500	499.002	26,200
#8	498.588	21,500	499.002	61,700	499.002	41,400	499.002	31,400
#9	498.588	29,800	499.002	69,000	499.002	47,000	499.002	34,500
#10	498.585	99,100	499.002	98,700	499.002	66,600	499.002	39,500
Average	498.712	33,510	499.002	56,220	499.002	39,000	499.002	25,290
Std. Dev.	0.107	24,999	0.000	17,705	0.000	10,833	0.000	7868
All executions								
Average	498.950	9109	499.002	16,971	499.002	13,938	499.002	7360
Std. Dev.	0.108	14,109	0.000	17,344	0.000	11,490	0.000	7982

The objective of the optimization problem is to minimize the average flux or power-peaking factor, f_p, of the proposed reactor, allowing the reactor to be sub-critical or super critical ($k_{eff} = 1.0 \pm 1\%$), for a given average flux ϕ_0. Let $\vec{D} = \{R_f, \Delta_c, R_e, E_1, E_2, E_3, M_f, M_c\}$ be the vector of design variables. Then, the optimization problem can be written as follows:

Minimize

$$f_p(\vec{D})$$

Subject to:

$$\phi(\vec{D}) = \phi_0; \tag{7}$$

$$0.99 \leq k_{eff}(\vec{D}) \leq 1.01; \tag{8}$$

$$\frac{dk_{eff}}{dV_m} > 0; \tag{9}$$

$$D_i^l \leq D_i \leq D_i^u, \quad i = 1, 2, \ldots, 6 \tag{10}$$

$$M_f = \{UO_2 \text{ or U-metal}\}; \tag{11}$$

$$M_c = \{\text{Zircaloy-2, Aluminum or Stainless-304}\}, \tag{12}$$

Table 3
Results for Rosenbrock's valley

Experiment	SGA		SA		GDA		PCA	
	Best	Iter.	Best	Iter.	Best	Iter.	Best	Iter.
Best results								
#1	0.000	900	0.000	100	0.000	200	0.000	100
#2	0.000	1600	0.000	200	0.000	200	0.000	200
#3	0.000	1900	0.000	400	0.000	400	0.000	200
#4	0.000	1900	0.000	400	0.000	400	0.000	300
#5	0.000	2100	0.000	600	0.000	700	0.000	300
#6	0.000	2100	0.000	700	0.000	800	0.000	300
#7	0.000	2200	0.000	800	0.000	800	0.000	300
#8	0.000	2400	0.000	1000	0.000	1200	0.000	400
#9	0.000	2600	0.000	1400	0.000	1200	0.000	500
#10	0.000	2900	0.000	1600	0.000	1300	0.000	500
Average	0.000	2060	0.000	720	0.000	720	0.000	310
Std. Dev.	0.000	552	0.000	494	0.000	416	0.000	129
Worst results								
#1	0.000	12,100	0.000	42,300	0.000	26,800	0.000	15,300
#2	0.000	12,400	0.000	42,700	0.000	27,100	0.000	16,900
#3	0.000	13,300	0.000	47,900	0.000	27,900	0.000	17,300
#4	0.000	13,600	0.000	50,300	0.000	28,900	0.000	17,900
#5	0.000	14,200	0.000	54,300	0.000	33,200	0.000	19,100
#6	0.000	16,300	0.000	55,000	0.000	35,700	0.000	20,300
#7	0.000	18,400	0.000	56,300	0.000	36,600	0.000	25,600
#8	0.000	21,500	0.000	57,000	0.000	41,800	0.000	27,900
#9	0.000	21,800	0.000	76,700	0.000	45,200	0.000	34,400
#10	0.000	29,700	0.000	82,200	0.000	49,500	0.000	37,300
Average	0.000	17,330	0.000	56,470	0.000	35,270	0.000	23,200
Std. Dev.	0.000	5619	0.000	13,251	0.000	8051	0.000	7763
All executions								
Average	0.000	7131	0.000	17,697	0.000	10,576	0.000	5953
Std. Dev.	0.000	4695	0.000	17,042	0.000	10,488	0.000	7233

where V_m is the moderator volume, and the superscripts l and u indicate, respectively, the lower and upper bounds (of the feasible range) for each design variable.

4.2. Optimization algorithms setup

The GA setup was the same as in Sacco et al. (2004), including the random seeds. As in Section 3, all the algorithms were set up for 100,000 iterations, so that the results were obtained with the same computational effort. The GDA's rain speed was set up using an initial rain level of -1.35 (a below-the-average solution) and an estimated final level of -1.28 (equal to the best results so far). For the SA, an initial temperature T_0 of -3.00 and a cooling schedule following

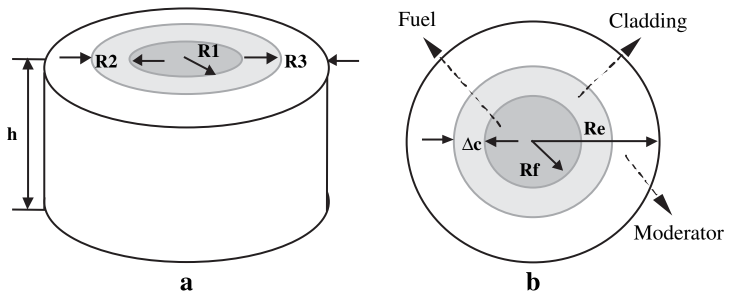

Fig. 9. (a) The nuclear reactor and (b) its typical cell.

Table 4
Parameters range

Parameter	Symbol	Range
Fuel radius (cm)	R_f	0.508–1.270
Cladding thickness (cm)	Δ_c	0.025–0.254
Moderator thickness (cm)	Δ_m	0.025–0.762
Enrichment of zone 1 (%)	E_1	2.0–5.0
Enrichment of zone 2 (%)	E_2	2.0–5.0
Enrichment of zone 3 (%)	E_3	2.0–5.0
Fuel material	M_f	{U-metal or UO_2}
Cladding material	M_c	{Zircaloy-2, Aluminum or Stainless Steel-304}

Eq. (2) with $\alpha = 0.95$ was used. The execution time for 100,000 iterations was 10 h 30 min in a Pentium IV 3.0 GHz PC with 1 Gb RAM. This time was independent of the method, as the bulk of the effort was taken up by the fitness evaluations, by the reactor physics code.

4.3. Reactor Physics Code

The HAMMER system (Suich and Honeck, 1967) was used for cell and diffusion equations' calculations. It performs a multigroup calculation of the thermal and epithermal flux distribution from the integral transport theory in a unit cell of the lattice.

$$\phi(r) = \int_V \frac{e^{-\Sigma_t |r-r'|}}{4\pi |r-r'|^2} S(r')\, d^3 r' \tag{13}$$

The integral transport equation for scalar flux $\phi(\vec{r})$ is solved for all sub-regions of the unit cell, being the neutron source $S(r)$ isotropic into the energy group under consideration. The transfer kernel in Eq. (13) is related to the collision probabilities for a flat isotropic source in the initial region. The solution is initially performed for a unit cell in an infinite lattice.

The integral transport calculation is followed by a multigroup Fourier transfer leakage spectrum theory in order to include the leakage effects in the previous calculation and to proceed with the multigroup flux–volume weighting.

Using the four group constants obtained from the mentioned procedure, a one-dimensional multi-region reactor calculation is performed. The diffusion equation is, then, solved to perform standard criticality calculation.

$$-\vec{\nabla} D_g(r)\vec{\nabla}\phi_g(r) + \Sigma_{t,g}(r)\phi_g(r) = \sum_{g'=1}^{4}\left[\frac{1}{k_{\mathrm{eff}}}\chi_g \Sigma_{fg'}(r) + \Sigma_{sg'g}(r)\right]\phi_{g'}(r) \tag{14}$$

The flux $\phi_g(r)$ is calculated assuming normalized source density. Eq. (14) is solved using the finite difference method and a computational mesh with constant spacing in the spatial coordinate.

4.4. Fitness function

The fitness function was developed in such a way that if all constraints are satisfied, it has the value of the average peak factor, f_p, otherwise, it is penalized proportionally to the discrepancy on the constraint. Such penalization factors should be set up by the expert, according to the requirements and the priorities of the problem, being weighted by the coefficients r_i, with $i = 1, 2, 3$.

$$f=\begin{cases} f_\text{p}, & \Delta k_\text{eff}\le 0.01;\Delta\phi\le 0.01\phi_0;\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}>0\\ f_\text{p}+r_1\Delta k_\text{eff}, & \Delta k_\text{eff}> 0.01;\Delta\phi\le 0.01\phi_0;\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}>0\\ f_\text{p}+r_2\Delta\phi, & \Delta k_\text{eff}\le 0.01;\Delta\phi> 0.01\phi_0;\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}>0\\ f_\text{p}+r_3\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}, & \Delta k_\text{eff}\le 0.01;\Delta\phi\le 0.01\phi_0;\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}<0\\ f_\text{p}+r_1\Delta k_\text{eff}+r_2\Delta\phi, & \Delta k_\text{eff}> 0.01;\Delta\phi> 0.01\phi_0;\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}>0\\ f_\text{p}+r_1\Delta k_\text{eff}+r_3\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}, & \Delta k_\text{eff}> 0.01;\Delta\phi\le 0.01\phi_0;\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}<0\\ f_\text{p}+r_2\Delta\phi+r_3\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}, & \Delta k_\text{eff}\le 0.01;\Delta\phi> 0.01\phi_0;\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}<0\\ f_\text{p}+r_1\Delta k_\text{eff}+r_2\Delta\phi+r_3\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}, & \Delta k_\text{eff}> 0.01;\Delta\phi> 0.01\phi_0;\dfrac{\Delta' k_\text{eff}}{\Delta V_\text{m}}<0 \end{cases} \tag{15}$$

4.5. Results

Tables 5 and 6 show the results for the power-peaking factor obtained by the PCA and GDA compared to those obtained by the SGA, SA, IGA and NGA for 10 independent experiments each with 50,000 and 100,000 iterations, respectively. The stopping criterion chosen was the number of iterations which enabled comparisons with previous efforts in the literature (Pereira et al., 1999; Pereira and Lapa, 2003; Sacco et al., 2004).

From these tables we can see that the PCA produced the best overall result and the GDA the best average and the lowest standard deviation. Also, both the PCA's and GDA's best results and averages at 50,000 iterations are better than the NGA's at 100,000. All SGA's executions drifted before 50,000 executions. Simulated Annealing was the method with the worst average for 50,000 executions, but it was better than the canonical Genetic Algorithm for 100,000 executions. As mentioned by Carter (1997), parameter-tuning was problematic for SA, especially in this problem where the search space is unknown.

The GDA and PCA performed well compared to the parallel Island Genetic Algorithm (IGA) of Pereira and Lapa (2003) which obtained the best results so far for this problem. With the same computational effort as the PCA's (4 islands of 50 individuals, 500 generations or 100,000 iterations), the IGA obtained 1.2784 as best result with an

Table 5
Comparison with the SGA, SA and the NGA for 50,000 iterations

Experiment	GA	SA	IGA	NGA	GDA	PCA
#1	1.3185	1.3449	N.A.	1.2916	*1.2806*	1.2849
#2	*1.3116*	1.3457	N.A.	1.3069	1.2913	1.2876
#3	1.3300	1.4055	N.A.	1.3003	1.2856	1.2964
#4	1.3294	1.3530	N.A.	*1.2874*	1.2909	1.2953
#5	1.3595	1.3770	N.A.	1.2956	1.2874	1.2829
#6	1.3562	1.3221	N.A.	1.3014	1.2845	*1.2791*
#7	1.3372	*1.3023*	N.A.	1.3190	1.2897	1.2975
#8	1.3523	1.3387	N.A.	1.3075	1.2953	1.2865
#9	1.3614	1.3380	N.A.	1.2974	1.2900	1.3010
#10	1.3467	1.3565	N.A.	1.3077	1.2930	1.2852
Average	1.3402	1.3484	N.A.	1.3015	1.2888	1.2896
Std. Dev.	0.0175	0.0283	N.A.	0.0093	0.0044	0.0073

Results for IGA were not available in Pereira and Lapa (2003).

Table 6
Comparison with the SGA, SA and the NGA for 100,000 iterations

Experiment	SGA	SA	IGA	NGA	GDA	PCA
#1	1.3185	1.3449	1.3322	1.2916	*1.2806*	1.2827
#2	*1.3116*	1.3390	1.2799	1.3069	1.2913	1.2876
#3	1.3300	1.3480	1.3378	1.3003	1.2856	1.2964
#4	1.3294	1.3530	1.2835	*1.2844*	1.2891	1.2874
#5	1.3595	1.3553	1.2810	1.2895	1.2863	1.2829
#6	1.3562	1.3221	1.2796	1.3014	1.2845	*1.2791*
#7	1.3372	*1.3023*	*1.2784*	1.2872	1.2897	1.2975
#8	1.3523	1.3387	1.3034	1.3050	1.2842	1.2865
#9	1.3614	1.3138	1.2989	1.2959	1.2895	1.2908
#10	1.3467	1.3565	1.3170	1.3077	1.2827	1.2845
Average	1.3402	1.3374	1.2992	1.2970	1.2864	1.2875
Std. Dev.	0.0175	0.0186	0.0228	0.0085	0.0035	0.0059

Results for IGA were taken from Pereira and Lapa (2003).

average of 1.2992, while the GDA's best was 1.2806 with an average of 1.2864 and the PCA's best was 1.2791 with an average of 1.2875.

Fig. 10 shows each algorithm's evolution for their best results, except for the IGA which was not available in Pereira and Lapa (2003).

Table 7 shows the best configurations obtained by the SGA (in Pereira et al., 1999) with 300 individuals until convergence, by the IGA (in Pereira and Lapa, 2003) with 400 individuals and 500 generations, by the NGA with 100 individuals (in Sacco et al., 2004) and 500 generations, and by GDA and PCA in 100,000 iterations, when applied exactly to the same problem. We note that the SGA runs by Pereira et al. produced a worse result even though a larger population was used, indicating the occurrence of genetic drift. The IGA result though comparable to that of the PCA required more computational effort.

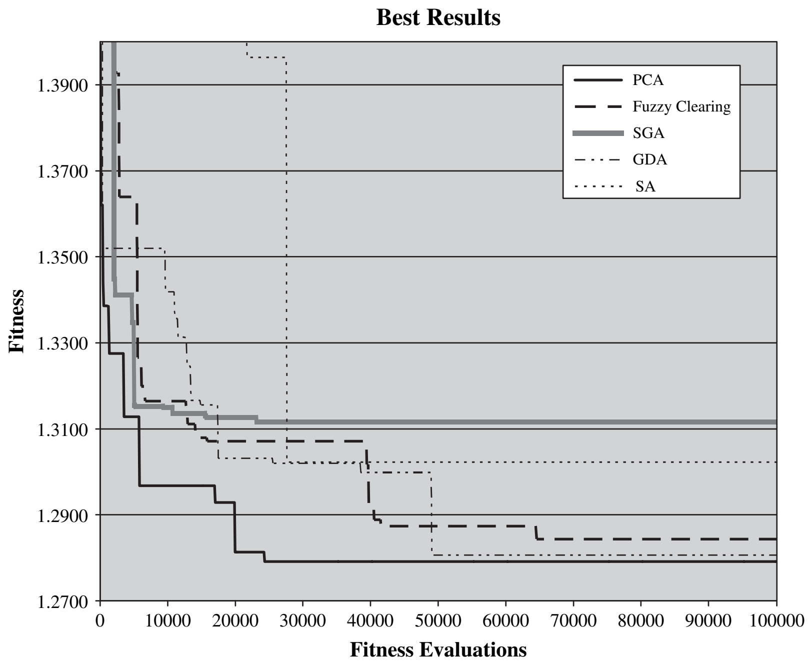

Fig. 10. Fitness evaluations vs. fitness for each algorithm's best configuration.

Table 7
Comparison with previously published best results

	SGA[a]	SA	IGA[b]	NGA[c]	GDA	PCA
Fitness	1.310	1.302	1.278	1.287	1.281	1.279
Minimum average peak factor	1.310	1.302	1.278	1.287	1.281	1.279
Average flux	8.02×10^{-5}	8.00×10^{-5}	8.08×10^{-5}	8.04×10^{-5}	7.95×10^{-5}	8.06×10^{-5}
k_{eff}	1.000	0.998	0.991	1.000	0.990	0.991
R_f (cm)	0.5621	0.5080	0.7661	0.5441	0.5913	0.5497
Δr (cm)	0.1770	0.0558	0.1857	0.1064	0.0638	0.1450
Δm (cm)	0.6581	0.5475	0.7569	0.5997	0.5992	0.6111
E_1 (%)	2.756	2.2072	2.6850	2.5906	2.1485	2.7953
E_2 (%)	4.032	2.5069	2.8268	2.7559	2.2585	2.9469
E_3 (%)	4.457	3.8932	4.8819	4.6220	3.8590	5.0000
M_f	U-metal	U-metal	U-metal	U-metal	U-metal	U-metal
M_c	Stainless-304	Stainless-304	Stainless-304	Stainless-304	Stainless-304	Stainless-304

[a] Pereira et al. (1999).
[b] Pereira and Lapa (2003).
[c] Sacco et al. (2004).

5. Conclusions

The comparative performances of PCA and GDA with the canonical Genetic Algorithm and its variants show that the first two algorithms are quite promising and could be applied to other optimization problems in the nuclear engineering field as, for instance, the nuclear core reload optimization problem (Poon and Parks, 1992).

The great advantage of PCA in relation to other optimization algorithms such as the Genetic Algorithm, Simulated Annealing or the Great Deluge Algorithm is that, other than the number of iterations, it does not require any additional parameters. Of course the performance of PCA and of the other algorithms depends crucially on the scaling of the fitness function, which is user-defined. The PCA can be applied to continuous or discrete optimization problems by just changing the perturbation function, while in Genetic Algorithms it is necessary to apply special operators for discrete optimization problems (Goldberg, 1989). Last but not least, the PCA is extremely easy to implement.

The Particle Collision Algorithm presented here is in its early stages. As further development, the local search mechanism could be improved, as a more intelligent mechanism could lead to significant gains in computational cost. Also, the perturbation and small-perturbation mechanisms should be studied in more detail. Another improvement would be a populational algorithm where the particles would interact to obtain better solutions, in a fashion similar to Particle Swarm Optimization (Kennedy and Eberhart, 1995).

Acknowledgements

Wagner F. Sacco is supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, Ministry of Science & Technology, Brazil) under postdoctoral grant 200223/2004-6. Claudio Márcio do Nascimento Abreu Pereira is supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, Ministry of Science & Technology, Brazil) under a research grant.

References

Aarts, E., Korst, J., 1989. Simulated Annealing and Boltzmann Machines. John Wiley and Sons, New York.

Baker, J.E., 1987. Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the Second International Conference on Genetic Algorithms and their Application. Hillsdale, NJ, p. 14.

Bezdek, J.C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York.

Bykov, Y., 2003. Time-predefined and Trajectory-based Search: Single and Multiobjective Approaches to Exam Timetabling. Ph.D. thesis, University of Nottingham, UK.

Cantu-Paz, E., 2000. Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Boston.

Carter, J.N., 1997. Genetic algorithms for incore fuel management and other recent developments in optimization. In: Advances in Nuclear Science and Technology, vol. 25. Plenum Press, New York, p. 113.

De Jong, K.A., 1975. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Doctoral dissertation, University of Michigan.

Duderstadt, J.J., Hamilton, L.J., 1976. Nuclear Reactor Analysis. John Wiley and Sons, New York.

Dueck, G., 1993. New optimization heuristics — the great deluge algorithm and record-to-record travel. Journal of Computational Physics 104, 86.

Easom, E.E., 1990. A Survey of Global Optimization Techniques. M. Eng. thesis, University of Louisville, Louisville, KY.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, Reading, MA.

Greffenstette, J.J., 1990. A User's Guide to GENESIS. Naval Research Laboratory, Washington, D.C.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. IV, p. 1942.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671.

Mahfoud, S.W., 1995. Niching Methods for Genetic Algorithms. Ph.D thesis, Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equations of state calculations by fast computing machines. Journal of Chemical Physics 21, 1087.

Pereira, C.M.N.A., Schirru, R., Martinez, A.S., 1999. Basic investigations related to genetic algorithms in core designs. Annals of Nuclear Energy 26, 173.

Pereira, C.M.N.A., Lapa, C.M.F., 2003. Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem. Annals of Nuclear Energy 30, 555.

Poon, P.W., Parks, G.T., 1992. Optimizing PWR reload core designs. In: Parallel Problem Solving from Nature 2, p. 371.

Rosenbrock, H.H., 1960. An automatic method for finding the greatest or least value of a function. Computer Journal 3, 175.

Sacco, W.F., Machado, M.D., Pereira, C.M.N.A., Schirru, R., 2004. The fuzzy clearing approach for a niching genetic algorithm applied to a nuclear reactor core design optimization problem. Annals of Nuclear Energy 31, 55.

Sacco, W.F., Oliveira, C.R.E., June 2005. A New Stochastic Optimization Algorithm based on Particle Collisions. 2005 ANS Annual Meeting. Transactions of the American Nuclear Society 92.

Shekel, J., 1971. Test functions for multimodal search techniques. In: Proceedings of the Fifth Princeton Conference on Information Science and Systems, Princeton, p. 354.

Suich, J.E., Honeck, H.C., 1967. The HAMMER System Heterogeneous Analysis by Multigroup Methods of Exponentials and Reactors. Savannah River Laboratory, Aiken, South Carolina.