2

An Adaptive Penalty Method for Genetic Algorithms in Constrained Optimization Problems

Helio J. C. Barbosa and Afonso C. C. Lemonge
Laboratório Nacional de Computação Científica, LNCC/MCT
Universidade Federal de Juiz de Fora, UFJF
Brazil

1. Introduction

Inspired by the Darwinian principles of evolution, genetic algorithms (GAs) (Holland, 1975, Goldberg, 1989) are widely used in optimization. GAs encode all the variables corresponding to a candidate solution (the phenotype) in a data structure (the genotype) and maintain a population of genotypes which is evolved mimicking Nature's evolutionary process. Individuals are selected for reproduction in a way that better performing solutions have a higher probability of being selected. The genetic material contained in the chromosome of such "parent" individuals is then recombined and mutated, by means of crossover and mutation operators, giving rise to offspring which will form a new generation of individuals. The process is repeated for a given number of generations or until some stopping criteria are met.

GAs are search algorithms which can be directly applied to unconstrained optimization problems, UOP (f, S), where one seeks for an element x belonging to the search space S, which minimizes (or maximizes) the real-valued objective function f. In this case, the GA usually employs a fitness function closely related to f. However, practical optimization problems often involve several types of constraints and one is led to a constrained optimization problem, COP (f, S, b), where an $x \in S$ that minimizes (or maximizes) f in S is sought with the additional requirement that the boolean function $b(x)$ (which includes all constraints of the problem) is evaluated as true. Elements in S satisfying all constraints are called feasible.

The straightforward application of GAs to COPs is not possible due to the additional requirement that a set of constraints must be satisfied. Difficulties may arise as the objective function may be undefined for some (or all) infeasible elements, and an informative measure of the degree of infeasibility of a given candidate solution may not be easily defined. Finally, for some real-world problems, the check for feasibility can be more expensive than the computation of the objective function itself.

As a result, several techniques have been proposed and compared in the literature in order to enable a GA to tackle COPs. Those techniques can be classified either as *direct* (feasible or

Source: Frontiers in Evolutionary Robotics, Book edited by: Hitoshi Iba, ISBN 978-3-902613-19-6, pp. 596, April 2008, I-Tech Education and Publishing, Vienna, Austria

interior), when only feasible elements in S are considered, or as *indirect* (exterior), when both feasible and infeasible elements are used during the search process.
Direct techniques include: a) the design of special *closed* genetic operators, b) the use of special decoders, c) repair techniques, and d) "death penalty".
In special situations, closed genetic operators (in the sense that when applied to feasible parents they produce feasible offspring) can be designed if enough domain knowledge is available (Shoenauer & Michalewicz, 1996). Special decoders (Koziel & Michalewicz, 1999) - that always generate feasible individuals from any given genotype have been devised, but no applications considering non-linear implicit constraints have been published.
Repair methods (Liepins & Potter, 1996, Orvosh & Davis, 1994) use domain knowledge in order to move an infeasible offspring into the feasible set. However, there are situations when it is very expensive, or even impossible, to construct such a repair operator, drastically reducing the range of applicability of repair methods. The design of efficient repair methods constitutes a formidable challenge, specially when non-linear implicit constraints are present.
Discarding any infeasible element generated during the search process ("death penalty") is common practice in non-populational optimization methods. Although problem independent, no consideration is made for the potential information content of any infeasible individual.
Summarizing, direct techniques are problem dependent (with the exception of the "death penalty") and actually of extremely reduced practical applicability.
Indirect techniques include: a) the use of Lagrange multipliers (Adeli & Cheng, 1994), which may also lead to the introduction of a population of multipliers and to the use of the concept of coevolution (Barbosa, 1999), b) the use of fitness as well as constraint violation values in a multi-objective optimization setting (Surry & Radcliffe, 1997), c) the use of special selection techniques (Runarsson & Yao, 2000), and d) "lethalization": any infeasible offspring is just assigned a given, very low, fitness value (Kampen et al., 1996).
For other methods proposed in the evolutionary computation literature see (Shoenauer & Michalewicz, 1996, Michalewicz & Shoenauer, 1996, Hinterding & Michalewicz, 1998, Koziel & Michalewicz, 1998, Kim & Myung, 1997), references therein, and the repository http://www.cs.cinvestav.mx/constraint/, maintained by C.A.C. Coello.
Methods to tackle COPs which require the knowledge of constraints in explicit form have thus limited practical applicability. This fact, together with simplicity of implementation are perhaps the main reasons why penalty techniques, in spite of their shortcomings, are the most popular ones.
Penalty techniques, originating in the mathematical programming community, range from simple schemes (like "lethalization") to penalty schemes involving from one to several parameters. Those parameters can remain constant (the most common case) or be dynamically varied along the evolutionary process according to an exogenous schedule or an adaptive procedure. Penalty methods, although quite general, require considerable domain knowledge and experimentation in each particular application in order to be effective.
In previous work (Barbosa & Lemonge, 2002, Lemonge & Barbosa, 2004), the authors developed a general penalty method for GAs which (i) handles inequality as well as equality constraints, (ii) does not require the knowledge of the explicit form of the constraints as a function of the decision/design variables, (iii) is free of parameters to be set by the user, (iv) can be easily implemented within an existing GA code and (v) is robust.

In this adaptive penalty method (APM), in contrast with approaches where a single penalty parameter is used, an adaptive scheme automatically sizes the penalty parameter corresponding to *each* constraint along the evolutionary process.
In the next section the penalty method and some of its traditional implementations within genetic algorithms are presented. In Section 3 the APM is revisited and some variants are proposed, Section 4 presents numerical experiments with several test-problems from the literature and the paper closes with some conclusions.

2. Penalty Methods

A standard constrained optimization problem in R^n can be written as the minimization of a given objective function $f(x)$, where $x \in R^n$ is the vector of design/decision variables, subject to inequality constraints $g_p(x) \geq 0,\ p = 1,2,\ldots,\overline{p}$ as well as equality constraints $h_q(x) = 0,\ q = 1,2,\ldots,\overline{q}$. Additionally, the variables may be subject to bounds $x_i^L \leq x_i \leq x_i^U$, but this type of constraint is trivially enforced in a GA.
Penalty techniques can be classified as *multiplicative* or *additive*. In the multiplicative case, a positive penalty factor $p(v(x),T)$ is introduced in order to amplify (in a minimization problem) the value of the fitness function of an infeasible individual. One would have $p(v(x),T) = 1$ for a feasible candidate solution x and $p(v(x),T) > 1$ otherwise. Also, $p(v(x),T)$ increases with the "temperature" T and with constraint violation. An initial value for the temperature is required as well as the definition of a schedule whereby T grows with the generation number. This type of penalty has received much less attention in the evolutionary computation community than the additive type. In the additive case, a penalty functional is added to the objective function in order to define the fitness value of an infeasible element. They can be further divided into: (a) *interior* techniques, when a barrier functional $B(x)$ -which grows rapidly as x approaches the boundary of the feasible domain- is added to the objective function

$$F_k(x) = f(x) + \frac{1}{k} B(x)$$

and (b) *exterior* techniques, where a penalty functional is introduced

$$F_k(x) = f(x) + kP(x) \tag{1}$$

such that $P(x) = 0$, if x is feasible, and $P(x) > 0$ otherwise (for minimization problems). In both cases, as $k \to \infty$, the sequence of minimizers of the UOP (F_k, S) converges to the solution of the COP (f, S, b).

At this point, it is useful to define the amount of violation of the j-th constraint by the candidate solution $x \in R^n$ as

$$v_j(x) = \begin{cases} |h_j(x)|, & \text{for an equality constraint,} \\ \max\{0, -g_j(x)\} & \text{otherwise} \end{cases}$$

It is also common to design penalty functions that grow with the vector of violations $v(x) \in R^m$ where $m = \overline{p} + \overline{q}$ is the number of constraints to be penalized. The most popular penalty function is given by

$$P(x) = k \sum_{j=1}^{m} (v_j(x))^{\beta} \tag{2}$$

where k is the penalty parameter and $\beta = 2$. Although it is easy to obtain the unconstrained problem, the definition of the penalty parameter k is usually a time-consuming problem-dependent trial-and-error process.

Le Riche et al. (1995) present a GA based on two-level of penalties, where two fixed penalty parameters k_1 and k_2 are used independently in two different populations. The idea is to create two sets of candidate solutions where one of them is evaluated with the parameter k_1 and the other with the parameter k_2. With $k_1 \neq k_2$, there are two different levels of penalization and there is a higher chance of maintaining feasible as well as infeasible individuals in the population and to get offspring near the boundary between the feasible and infeasible regions. The strategy can be summarized as: (i) create $2 \times pop$ individuals randomly (pop is the population size); (ii) evaluate each individual considering each penalty parameter and create two ranks; (iii) combine the two ranks in a single one with size pop; (iv) apply genetic operators; (v) evaluate new offspring ($2 \times pop$), using both penalty parameters and repeat the process.

Homaifar et al. (1994) proposed a penalty strategy with multiple coefficients for different levels of violation of each constraint. The fitness function is written as

$$F(x) = f(x) + \sum_{j=1}^{m} k_{ij} (v_j(x))^2$$

where i denotes one of the l levels of violation defined for the $j-$th constraint. This is an attractive strategy because, at least in principle, it allows for a good control of the penalization process. The weakness of this method is the large number, $m(2l+1)$, of parameters that must be set by the user for each problem.

Joines & Houck (1994) proposed that the penalty parameters should vary dynamically along the search according to an exogenous schedule. The fitness function $F(x)$ was written as in (1) and (2) with the penalty parameter, given by $k = (C \times t)^{\alpha}$, increasing with the generation number t. They used both $\beta = 1, 2$ and suggested the values $C = 0.5$ and $\alpha = 2$.

Powell & Skolnick (1994), proposed a method of superiority of feasible points where each candidate solution is evaluated by the following expression:

$$F(x) = f(x) + r\sum_{j=1}^{m} v_j(x) + \theta(t, x)$$

where r is a constant. The main assumption is that any feasible solution is better than any infeasible solution. This assumption is enforced by a convenient definition of the function $\theta(t, x)$. This scheme can be modified by the introduction of the tournament selection proposed in (Deb, 2000), coupled with the fitness function:

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

where f_{max} is the function value of the worst feasible solution. As observed in the expression above, if a solution is infeasible the value of its objective function is not considered in the computation of the fitness function, instead, f_{max} is used. However, Deb's constraint handling scheme (Deb, 2000) -which also uses niching and a controlled mutation- works very well for his real-coded GA, but not so well for his binary-coded GA.

Among the many suggestions in the literature (Powell & Skolnick, 1993, Michalewicz & Shoenauer, 1996, Deb, 2000) some of them -more closely related to the work presented here- will be briefly discussed in the following.

2.1 Adaptive penalties

A procedure where the penalty parameters change according to information gathered during the evolution process was proposed by Bean & Hadj-Alouane (Bean & Alouane, 1992). The fitness function is again given by (1) and (2) but with the penalty parameter $k = \lambda(t)$ adapted at each generation by the following rules:

$$\lambda(t+1) = \begin{cases} (1\beta_1)\lambda(t), & \text{if } b^i \in F \text{ for all } t - g + 1 \le i \le t \\ \beta_2 \lambda(t), & \text{if } b^i \notin F \text{ for all } t - g + 1 \le i \le t \\ \lambda(t) & \text{otherwise} \end{cases}$$

where b^i is the best element at generation i, F is the feasible region, $\beta_1 \neq \beta_2$ and $\beta_1, \beta_2 > 1$. In this method, the penalty parameter of the next generation $\lambda(t+1)$ decreases when all best elements in the last g generations are feasible, increases if all best elements are infeasible, and otherwise remains without change. The fitness function is written as:

$$F(x)=f(x)+\lambda(t)\left[\sum_{i=1}^{n} g_i^2(x)+\sum_{j=1}^{p}\left|h_j(x)\right|\right]$$

Schoenauer & Xanthakis (1993), presented a strategy that handles constrained problems in stages: (i) initially, a randomly generated population is evolved considering only the first constraint until a certain percentage of the population is feasible with respect to that constraint; (ii) the final population of the first stage of the process is used in order to optimize with respect to the second constraint. During this stage, the elements that had violated the previous constraint are removed from the population, (iii) the process is repeated until all the constraints are processed. This strategy becomes less attractive as the number of constraints grows and is potentially dependent on the order in which the constraints are processed.
The method proposed by Coit et al. (1996), uses the fitness function:

$$F(x) = f(x) + (F_{feas}(t) - F_{all}(t))\sum_{j=1}^{m}(v_j(x)/v_j(t))^{\alpha}$$

where $F_{all}(t)$ corresponds to the best solution (without penalty), until the generation t, F_{feas} corresponds to the best feasible solution, and α is a constant. The fitness function is also written as:

$$F(x) = f(x) + (F_{feas}(t) - F_{all}(t))\sum_{j=1}^{m}(v_j(x)/NFT(t))^{\alpha}$$

where NFT (Near Feasibility Threshold) defines the threshold distance, set by the user, between the feasible and infeasible regions of the search space.
A variation of the method proposed by Coit et al. (1996) is presented in (Gen & Gheng, 1996a), where the fitness function is defined as:

$$F(x) = f(x)\left[1-\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\Delta b_i(x)}{b_i}\right)\right]$$

and $\Delta b_i(x) = \max[0, g_i(x) - b_i]$ refers to the constraint violation. Based upon the same idea, Gen and Cheng (1996b), proposed an adaptive penalty scheme where the fitness function is written as:

$$F(x) = f(x)\left[1 - \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\Delta b_i(x)}{\Delta b_i^{max}}\right)\right]$$

where $\Delta b_i^{max} = \max\left[\varepsilon, \Delta b_i(x)\right]$ is the maximum violation of the constraint i and ε is a small positive value to avoid dividing by zero.
Hamida & Schoenauer (2000), proposed an adaptive scheme using: (i) a function of the proportion of feasible individuals in the population, (ii) a seduction/selection strategy to mate feasible and infeasible individuals, and (iii) a selection scheme to give advantage for a given number of feasible individuals.

3. The Adaptive Penalty Method

The adaptive penalty method (APM) was originally introduced in (Barbosa & Lemonge, 2002). In (Barbosa & Lemonge, 2003), the behavior of the penalty parameters was studied and further tests were performed. Finaly, in (Lemonge & Barbosa, 2004.), a slight but important modification was introduced leading to its present form. The method does not require any type of user defined penalty parameter and uses information from the population, such as the average of the objective function and the level of violation of each constraint during the evolution.

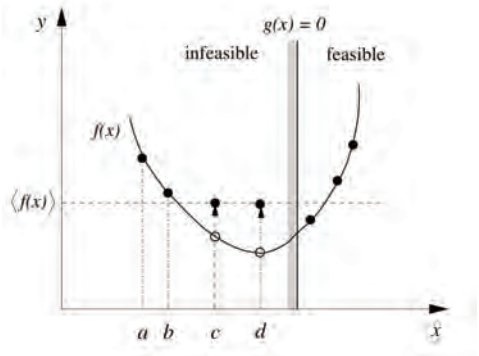

Figure 1. A pictorial description of the function $\overline{f}$

The fitness function proposed in APM is written as:

$$F(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible} \\ \overline{f}(x) + \sum_{j=1}^{m} k_j v_j(x) & \text{otherwise} \end{cases}$$

where

$$\overline{f}(x) = \begin{cases} f(x), & \text{if } f(x) > \langle f(x) \rangle, \\ \langle f(x) \rangle & \text{otherwise} \end{cases} \tag{3}$$

and $\langle f(x)\rangle$ is the average of the objective function values in the current population. In the Figure 1 feasible as well as infeasible solutions are shown. Among the 4 infeasible solutions, the individuals *c* and *d* have their objective function values (represented by open circles) less than the average objective function and, according to the proposed method, have $\overline{f}$ given by $\langle f(x)\rangle$. The solutions *a* and *b* have objective function values which are worst than the population average and thus have $\overline{f}(x) = f(x)$.
The penalty parameter is defined at each generation by:

$$k_j = |\langle f(x)\rangle| \frac{\langle v_j(x)\rangle}{\sum_{l=1}^{m}[\langle v_l(x)\rangle]^2} \tag{4}$$

and $\langle v_l(x)\rangle$ is the violation of the l-th constraint averaged over the current population. Denoting by pop the population size, one could also write

$$k_j = \frac{|\sum_{i=1}^{pop} f(x^i)|}{\sum_{l=1}^{m}\left[\sum_{i=1}^{pop} v_l(x^i)\right]^2} \sum_{i=1}^{pop} v_j(x^i) \tag{5}$$

The idea is that the values of the penalty coefficients should be distributed in a way that those constraints which are more difficult to be satisfied should have a relatively higher penalty coefficient.
Besides been tested by the authors, the APM has also been used by other researchers in different fields (Zavislak, 2004, Gallet, 2005, Obadage & Hampornchai, 2006).

4. Variants of the APM

The adaptive penalty method, as originally proposed, (i) computes the constraint violations v_j in the current population, and (ii) updates all penalty coefficients k_j at every generation. As a result, those coefficients undergo (potentially large) variations at every generation as shown in (Barbosa & Lemonge, 2003).
Despite its success, questions naturally arise concerning, for instance, the desirability of reducing the amplitude of those oscilations, and the possibility of reducing the computer time without compromising the quality of the results.
In the following, some possible variants of the APM will be considered.

4.1 The sporadic APM

One first variant of the APM corresponds to: (i) compute the constraint violations v_j in the current population, (ii) update the penalty coefficients k_j, but (iii) keep the penalty coefficients k_j fixed for f generations.

The gains in computer time are obvious: step (i) is performed only every f generations. Of course, one should investigate if convergence is negatively affected.

4.2 The sporadic APM with constraint violation accumulation

A second variant considered here corresponds to: (i) accumulate the constraint violations v_j for f generations, (ii) update the penalty coefficients k_j, and (iii) keep the penalty coefficients k_j fixed for f generations

The gains in computer time are very small, but the oscilations in the k_j values can be reduced.

4.3 The APM with monotonic penalty coefficients

As the penalty coefficients should (at least theoretically) grow indefinitely, another variant corresponds to the monotonic APM: no k_j is allowed to have its value reduced along the evolutionary process:

4.4 The APM with damping

Finaly, one can think of reducing the oscilations in k_j by using a weighted average between the current value of k_j and the new value predicted by the method:

$$k_j^{(new)} = \theta k_j^{(new)} + (1-\theta) k_j^{current}$$

where $\theta \in [0,1]$.

4.5 A compact notation

A single notation can be introduced in order to describe the whole family of adaptive penalty techniques.

The numerical parameters involved are:

- The frequency f (measured in number of generations) of penalty parameter updating,
- The weight parameter $\theta \in [0,1]$ used in the update formula for the penalty parameter.

Two additional features are: (i) constraint violations can alternatively be accumulated for all the f generations, or can be computed only every f generations, and (ii) the penalty parameter is free to vary along the process, or monotonicity will be enforced.

The compact notation APM(f^{acc} , θ_m) indicates a member of the family of methods where the penalty coefficients are updated every f generations, based on constraint violations which are accumulated (superscript acc) over those f generations, using the weight parameter θ, and enforcing monotonicity (subscript m). Thus, the original APM corresponds to APM(1,1).
The five members of the APM family considered for testing here are:

Variant	Description
1	• The original APM: APM(1,1)
2	• The sporadic APM: APM(f ,1)
3	• The sporadic APM with constraint violation accumulation: APM(f^{acc} ,1)
4	• The sporadic APM with constraint violation accumulation and smoothing: APM(f^{acc} , θ)
5	• The sporadic APM with constraint violation accumulation, smoothing, and monotonicity enforcement: APM(f^{acc} , θ_m)

In the experiments performed, the values $f = 100$ and $f = 500$ are used when 1000 and 5000 generations are employed, respectively. Also, $\theta = 0.5$ was adopted.
Several examples from the literature are considered in the next section in order to test the performance of the variants considered with respect to the original APM.

5. Numerical experiments

Our baseline binary-coded generational GA uses a Gray code, rank-based selection, and elitism (the best element is always copied into the next generation along with 1 copy where one bit has been changed). A uniform crossover operator was applied with probability equal to 0.9 and a mutation operator was applied bit-wise to the offspring with rate $p_m = 0.004$.

5.1 Test 1 - The G-Suite

The first set of experiments uses the well known G1-G11 suite of test-problems. A complete description of each problem can be found, for example, in (Lemonge, & Barbosa, 2004). The G-Suite, repeatedly used as a test-bed in the evolutionary computation literature, is made up of different kinds of functions and involves constraints given by linear inequalities, nonlinear equalities and nonlinear inequalities. For the eleven problems, the population size is set to 100, 25 independent runs were performed, with the maximum number of generations equal to 1000 and 5000, and each variable was encoded with 25 bits. For the variant 5, the value of θ is set to 0.5. The Tables 1 and 2 present the values of the best, mean and the worst solutions found for each function with respect to the five variants of the APM. The values presented in **boldface** correspond to the better solution among the variants.

Test-problem	Best-known	Variant	Best	Mean	Worst
g01	-15	1	-14.9996145	-14.9845129	-14.8267331
		2	**-14.9997983**	**-14.9988652**	-14.9939536
		3	-14.9997216	-14.9876210	-14.8305302
		4	-14.9997582	-14.9986873	-14.9960541
		5	-14.9996321	-14.9979965	-14.9891883
g02	-0.8036191	1	-0.7789661	-0.6996669	-0.5728089
		2	-0.7873126	-0.7250099	-0.5998890
		3	**-0.7925228**	**-0.7255508**	-0.6244844
		4	-0.7846081	-0.7191555	-0.6307588
		5	-0.7838283	-0.7097919	-0.5855941
g03	-1.0005001	1	**-0.9972458**	**-0.7779740**	-0.5027824
		2	-0.9361403	-0.4968387	-0.0368749
		3	-0.9803577	-0.4463771	-0.1012515
		4	-0.9682280	-0.4168279	-0.0212638
		5	-0.7666858	-0.3919809	-0.1462019
g04	-30665.538671	1	**-30665.3165485**	-30578.5496875	-30387.6140423
		2	-30664.6156191	-30578.3703906	-30458.6292214
		3	-30663.9053552	-30577.5611718	-30432.1269937
		4	-30664.4464089	-30573.7182031	-30369.9518043
		5	-30662.5018601	**-30591.6589843**	-30358.6826419
g05	5126.4967140	1	5127.3606448	5343.2514134	5993.0123939
		2	5127.0852075	5341.0769391	5935.3076090
		3	**5126.7785185**	5323.8655104	5935.3019471
		4	5127.0852075	**5290.0768500**	5935.3076090
		5	5128.8444266	5378.8769735	6102.9022396
g06	-6961.8138755	1	-6957.5403309	**-6913.0707583**	-6868.6591021
		2	-6951.1888908	-6901.2575390	-6705.1840698
		3	-6958.1031284	-6879.5979882	-6296.5426371
		4	-6954.9453586	-6906.1789453	-6797.2374636
		5	**-6961.4475438**	-6805.2293359	-5237.1477535
g07	24.3062090	1	24.7768086	**27.7708738**	35.6097627
		2	24.8049743	28.0120724	28.0120724
		3	25.0308511	28.4575817	44.1698225
		4	**24.5450354**	27.8486413	36.8722275
		5	24.7255055	29.8150354	63.5219816
g08	-0.0958250	1	**-0.0958250**	-0.08768981	-0.0258067
		2	**-0.0958250**	-0.06476799	-0.0013001
		3	**-0.0958250**	-0.62599863	-0.0147534
		4	**-0.0958250**	**-0.65938715**	-0.0255390
		5	**-0.0958250**	-0.53489946	0.0123813

Test-problem	Best-known	Variant	Best	Mean	Worst
g09	680.6300573	1	680.7376315	682.0270142	690.4830475
		2	680.7722707	681.5344751	685.8184957
		3	680.7343755	**681.4195117**	683.1598659
		4	680.8243208	681.8727295	687.0344127
		5	**680.6814564**	681.4697436	683.2056847
g10	7049.2480205	1	**7070.5636522**	**8063.2916015**	8762.119390
		2	7237.0250892	10056.5608887	14688.684065
		3	7191.9045654	10165.4067383	14733.840629
		4	7217.4084043	9607.9555664	13715.794540
		5	7117.7173709	9322.0432129	12645.139817
g11	0.7499000	1	0.7523536	**0.8585980**	0.9932359
		2	0.7521745	0.8878919	0.9992905
		3	**0.7521730**	0.8879267	0.9992905
		4	0.7521745	0.8879346	0.9991157
		5	0.7501757	0.8822223	0.9954779

Table 1. Results of the G-Suite using 1000 generations

Test-problem	Best-known	Variant	Best	Mean	Worst
g01	-15	1	-14.9999800	-14.95263298	-14.6373581
		2	**-14.9999914**	-14.99962558	-14.9969186
		3	-14.9999826	-14.99953311	-14.9920359
		4	-14.9999910	-14.99998634	-14.9999264
		5	-14.9999915	**-14.9999189**	-14.9996256
g02	-0.8036191	1	-0.8015561	-0.77248005	-0.7261164
		2	-0.8025720	-0.76760471	-0.7328427
		3	-0.8023503	-0.77583072	-0.7401626
		4	**-0.8030480**	-0.77158681	-0.7752280
		5	-0.8025918	**-0.77851344**	-0.7306644
g03	-1.0005001	1	**-1.0004896**	**-1.00040364**	-0.9999571
		2	-1.0004634	-0.98604559	-0.6746467
		3	-1.0004630	-0.97507574	-0.3823454
		4	-1.0004602	-0.97900237	-0.5014565
		5	-1.0004553	-0.99997359	-0.9962939
g04	-30665.538671	1	-30665.523774	-30665.094688	-30661.761001
		2	-30665.536925	-30664.093203	-30643.260211
		3	**-30665.537505**	**-30665.365000**	-30664.505410
		4	-30665.533257	-30664.861094	-30660.745418
		5	-30665.520939	-30664.819062	-30660.360061

Test-problem	Best-known	Variant	Best	Mean	Worst
g05	5126.4967140	1	5127.3606448	**5321.5713820**	5993.0113312
		2	**5126.7738829**	5323.8496730	5935.3030129
		3	**5126.7738829**	5323.8495456	5935.3030129
		4	**5126.7738829**	5323.8496943	5935.3030129
		5	5128.8443865	5385.9205375	6102.8994189
g06	-6961.8138755	1	-6961.7960836	-6742.3431201	-1476.0027566
		2	-6961.7960836	**-6961.7763086**	-6961.7592083
		3	-6961.7960836	-6961.7720312	-6961.7592083
		4	-6961.7960836	**-6961.7763086**	-6961.7592083
		5	**-6961.7960837**	-6961.7752278	-6961.7592083
g07	24.3062090	1	24.4162533	26.4573583	30.7900065
		2	24.6482626	27.2199523	39.8419615
		3	24.5543275	26.9851276	38.2220049
		4	24.7315254	27.8377833	51.1675542
		5	**24.4148575**	**26.3590182**	29.5413519
g08	-0.0958250	1	**-0.0958250**	**-0.0876899**	-0.0258094
		2	**-0.0958250**	-0.0565442	-0.0010461
		3	**-0.0958250**	-0.0649342	-0.0010460
		4	**-0.0958250**	-0.0660025	-0.0204120
		5	**-0.0958250**	-0.0632326	-0.0197011
g09	680.6300573	1	**680.6334486**	**680.7850952**	681.1733250
		2	680.6673020	680.8470361	681.7406035
		3	680.7122589	680.8832129	681.3188377
		4	680.6505355	680.8678125	681.3625388
		5	680.6527662	680.8803711	681.6149132
g10	7049.2480205	1	7205.1435740	**8392.399951**	10349.909364
		2	7064.1563375	10108.975371	15407.413766
		3	**7062.6060743**	9953.722324	14200.173044
		4	7064.5428438	8554.065429	13156.5681862
		5	7064.3344362	10079.815684	14387.7769407
g11	0.7499000	1	0.7523536	**0.8556172**	0.9468393
		2	0.7521745	0.8834432	0.9903447
		3	0.7521745	0.8834432	0.9903447
		4	0.7521745	0.8845256	0.9937399
		5	**0.7501757**	0.8761453	0.9944963

Table 2. Results of the G-Suite using 5000 generations

Observing the results presented in Tables 1 and 2 one can see that the original APM found more often the best solutions. Among the new variants, the variants 3 and 5 were the most successful. As the performance of variant 4 was not as good as that of variants 3 and 5, one may suspect that, at least for this set of problems, the inclusion of the damping procedure was detrimental.

The graphics from the figures 2 to 25 show the values of the coefficients kj along the 1000 generations performed for the functions *g06*, *g09*, *g04* and *g01*. For the Variant 1 are drawn

two graphics where the second one correponds to a logarithmic scale for the coefficient k_j in the *y*-axis.

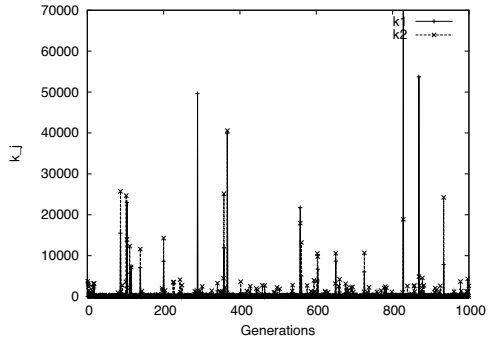

Figure 2. Variant 1 – Test-problem *g06*

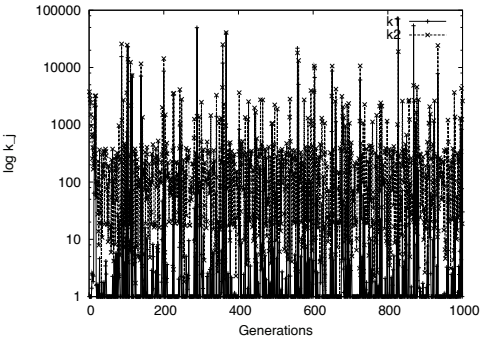

Figure 3. Variant 1 – Test-problem *g06* (log scale for *y*-axis)

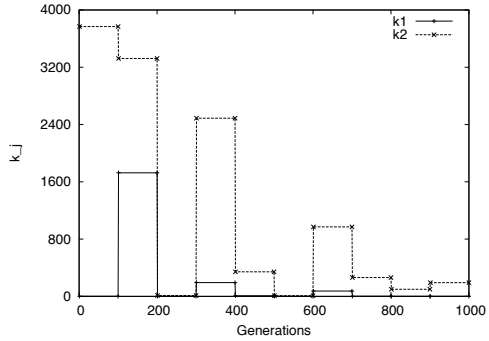

Figure 4. Variant 2 – Test-problem *g06*

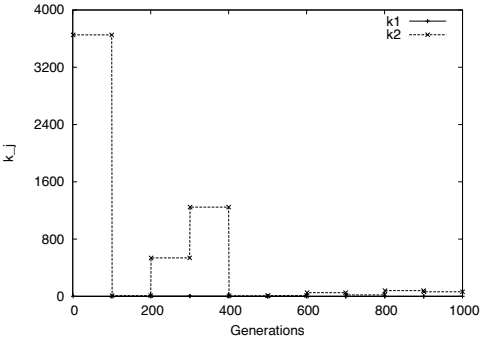

Figure 5. Variant 3 – Test-problem *g06*

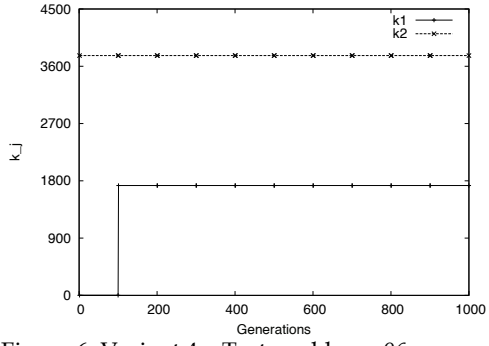

Figure 6. Variant 4 – Test-problem *g06*

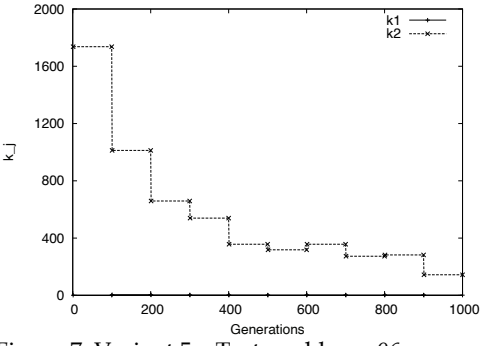

Figure 7. Variant 5 – Test-problem *g06*

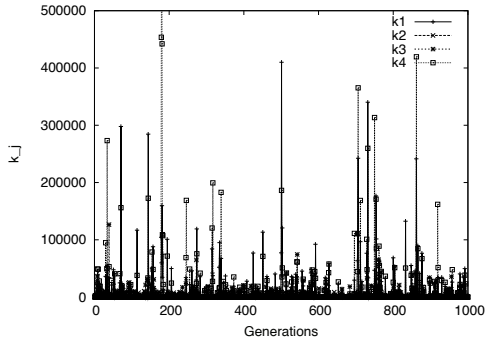

Figure 8. Variant 1 – Test-problem *g09*

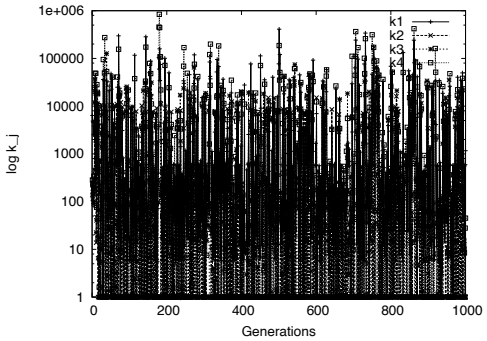

Figure 9. Variant 1 – Test-problem *g09* (log scale for *y*-axis)

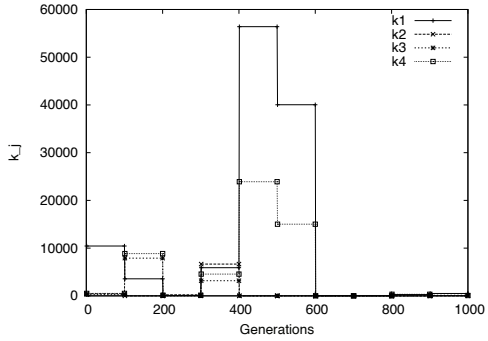

Figure 10. Variant 2 – Test-problem *g09*

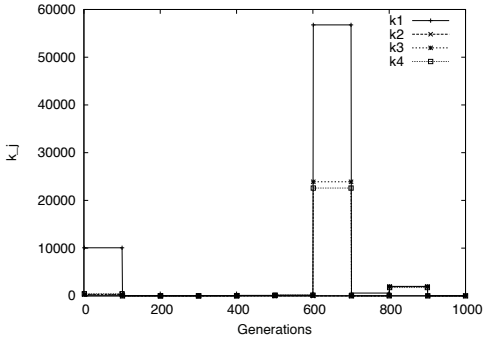

Figure 11. Variant 3 – Test-problem *g09*

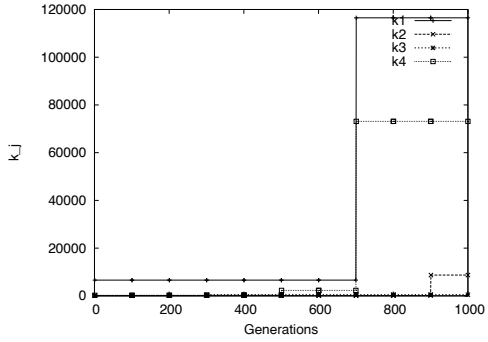

Figure 12. Variant 4 – Test-problem *g09*

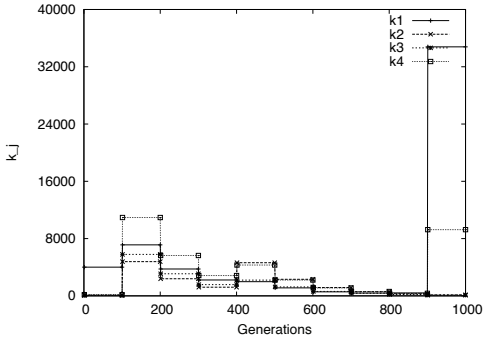

Figure 13. Variant 5 – Test-problem *g09*

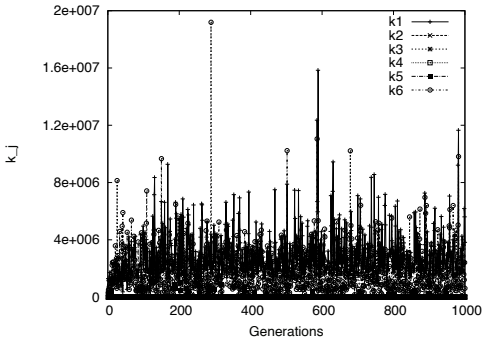

Figure 14. Variant 1 – Test-problem *g04*

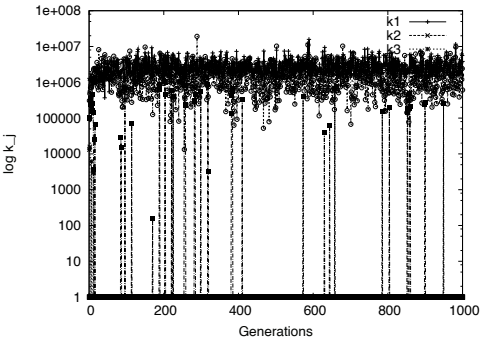

Figure 15. Variant 1 – Test-problem *g04* (log scale for *y*-axis)

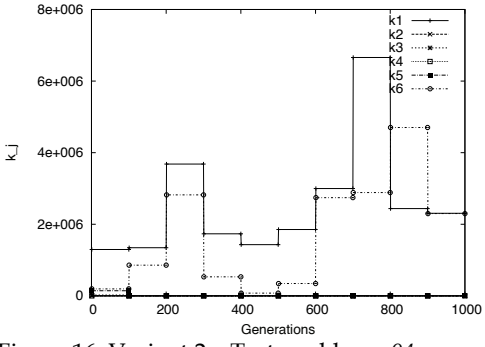

Figure 16. Variant 2 – Test-problem *g04*

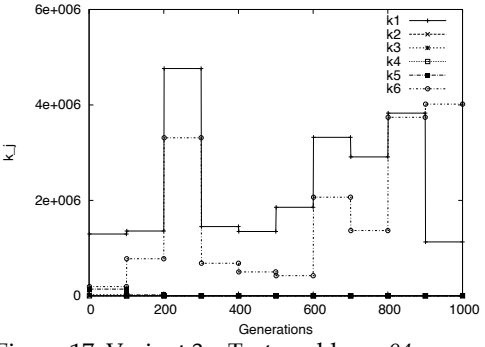

Figure 17. Variant 3 – Test-problem *g04*

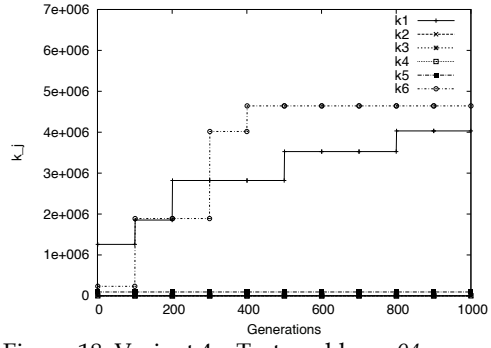

Figure 18. Variant 4 – Test-problem *g04*

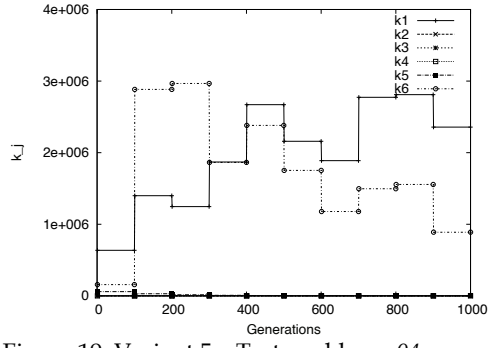

Figure 19. Variant 5 – Test-problem *g04*

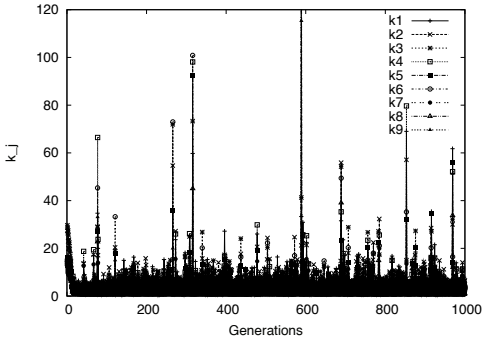

Figure 20. Variant 1 – Test-problem *g01*

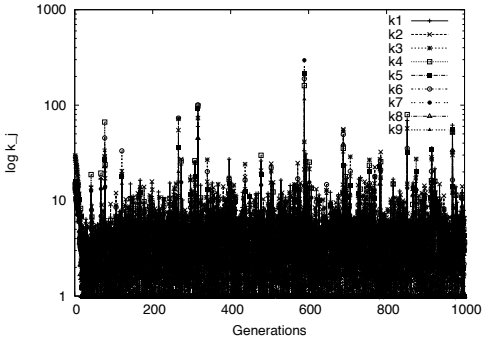

Figure 21. Variant 1 – Test-problem *g01* (log scale for *y*-axis)

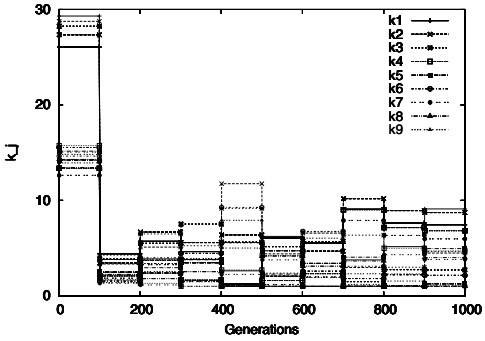

Figure 22. Variant 2 – Test-problem *g01*

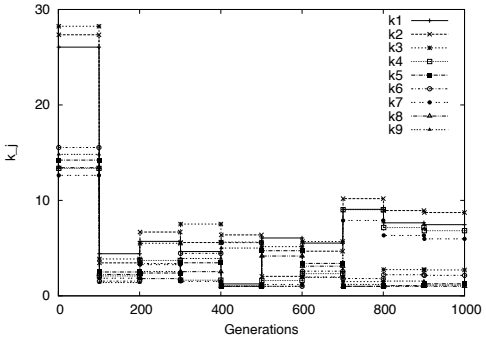

Figure 23. Variant 3 – Test-problem *g01*

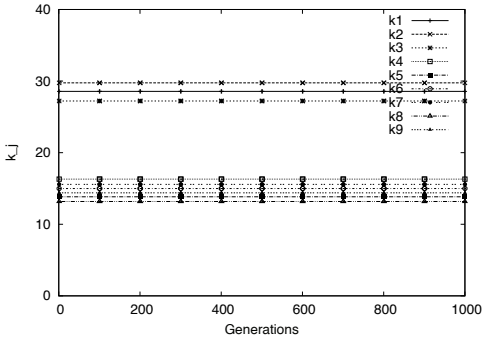

Figure 24. Variant 4 – Test-problem *g01*

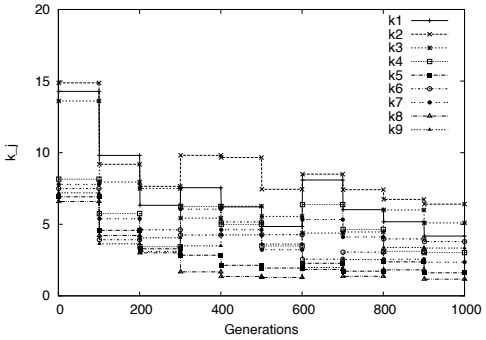

Figure 25. Variant 5 – Test-problem *g01*

5.2 Test 2 - The pressure vessel design

This problem (Sandgren, 1998, Kannan & Kramer, 1995, Deb, 1997, Coello, 2000) corresponds to the weight minimization of a cylindrical pressure vessel with two spherical heads. The objective function involves four variables: the thickness of the pressure vessel (T_s), the thickness of the head (T_h), the inner radius of the vessel (R) and the length of the cylindrical component (L). Since there are two discrete variables (T_s and T_h) and two continuous variables (R and L), one has a nonlinearly constrained mixed discrete-continuous optimization problem. The bounds of the design variables are $0.0625 \leq T_s, T_h \leq 5$ (in constant steps of 0.0625) and $10 \leq R, L \leq 200$. The design variables are given in inches and the weight is written as:

$$W(T_s, T_h, R, L) = 0.6224 T_s T_h R + 1.7781 T_h R^2 + 3.1661 T_s^2 L + 19.84 T_s^2 R$$

to be minimized subject to the constraints

$$g_1(T_s, R) = T_s - 0.0193R \geq 0 \qquad g_2(T_h, R) = T_h - 0.00954R \geq 0$$

$$g_3(R, L) = \pi R^2 L + 4/3\pi R^3 - 1{,}296{,}000 \geq 0 \quad g_4(L) = -L + 240 \geq 0$$

The first two constraints establish a lower bound to the ratios T_s/R and T_h/R, respectively. The third constraint corresponds to a lower bound for the volume of the vessel and the last one to an upper bound for the length of the cylindrical component.

The Table 3 presents the results of the pressure vessel design using 1000 generations. Two identical results were found for the best value of the final weight (6059.7145293) with respect to the APM variants 2 and 3. Using 5000 generations, the same value of the final weight was found four times as shown in the Table 4. Except for the original APM (variant 1), all variants were able to find the best solutions.

Variant	best	median	Mean	std	worst	frun
1	6059.7151671	**6227.8170385**	6474.4169921	4.53E+02	7509.8518229	25
2	**6059.7145293**	6384.0497026	**6447.2068164**	4.19E+02	7340.3461505	25
3	**6059.7145293**	6410.2945786	6481.3246679	4.29E+02	7544.4928499	25
4	6059.7187531	6371.1972928	6467.6047851	4.30E+02	7544.5662019	25
5	6059.7151671	6376.8444814	6502.6069921	4.37E+02	7544.4928499	25

Table 3. Results of the pressure vessel design using 1000 generations

variant	best	median	Mean	std	worst	frun
1	6059.7903738	6525.0826625	6571.3568945	4.79E+02	7333.0935211	25
2	**6059.7145293**	6370.7942228	6448.2773437	4.21E+02	7334.9819158	25
3	**6059.7145293**	6372.4349034	6445.8280859	4.16E+02	7340.8828459	25
4	**6059.7145293**	6370.7970442	6466.9971289	4.03E+02	7334.2645178	25
5	**6059.7145293**	**6370.7798337**	**6445.3129101**	4.58E+02	7332.9812503	25

Table 4. Results of the pressure vessel design using 5000 generations

5.3 Test 3 - The cantilever beam design

This test problem (Erbatur et al., 2000) corresponds to the minimization of the volume of the cantilever beam subject to the load P, equal to 50000 N, at its tip. There are 10 design variables corresponding to the height (H_i) and width (B_i) of the rectangular cross-section of each of the five constant steps. The variables B_1 and H_1 are integer, B_2 and B_3 assume discrete values to be chosen from the set $\{2.4,2.6,2.8,3.1\}$, H_2 and H_3 are discrete and chosen from the set $\{45.0,50.0,55.0,60.0\}$ and, finally, B_4, H_4, B_5, and H_5 are continuous. The variables are given in centimeters and the Young's modulus of the material is equal to 200 GPa. The volume of the beam is given by

$$V(H_i,B_i) = 100\sum_{i=1}^{5} H_i B_i$$

subject to:

$$g_i(H_i,B_i) = \sigma_i \le 14000 N/cm^2 \quad i = 1,\ldots,5$$
$$g_{i+5}(H_i,B_i) = H_i/B_i \le 20 \quad i = 1,\ldots,5$$
$$g_{11}(H_i,B_i) = \delta \le 2.7cm$$

where δ is the tip deflection of the beam in the vertical direction.

The Table 5 shows the results for the cantilever beam design when 1000 generations were used. The best final volume found (64842.0265521), was achieved by the third variant of the APM variant. Using 5000 generations, the variant 4 provided the best solution among the variants, with a final volume equal to 64578.1955255 as shown in the Table 6.

variant	best	median	mean	std	worst	frun
1	65530.7729044	68752.485343	**70236.739375**	3.59E+03	78424.35756	25
2	64991.8776321	68990.278258	70642.622188	4.47E+03	84559.93461	25
3	**64842.0265521**	**68662.595134**	70312.300781	4.39E+03	78015.93963	25
4	64726.0899470	68924.741220	71383.862812	5.60E+03	85173.95180	25
5	65281.3580659	69142.397989	69860.818906	3.66E+03	78787.491031	25

Table 5. Results of the cantilever beam design using 1000 generations

ariant	best	median	mean	std	worst	frun
1	64578.1959540	68293.437745	67153.214844	2.00E+03	72079.947779	25
2	64578.1957397	**64965.027481**	**66161.232500**	1.77E+03	68391.603479	25
3	64578.1959540	64992.041635	66704.774531	3.13E+03	79124.364956	25
4	**64578.1955255**	68201.363366	66682.130313	1.86E+03	68722.669573	25
5	64578.1957397	65015.052899	66450.535156	1.89E+03	69294.203883	25

Table 6. Results of the cantilever beam design using 5000 generations

5.4 Test 4 - The Tension/Compression Spring design

This example corresponds to the minimization of the volume V of a coil spring under a constant tension/compression load. There are three design variables: the number $x_1 = N$ of active coils of the spring, the winding diameter $x_2 = D$ and the wire diameter $x_3 = d$. The volume of the coil is written as:

$$V(x) = (x_3 + 2)x_2 x_1^2$$

and is subject to the constraints:

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71875 x_1^4} \le 0 \qquad g_2(x) = \frac{x_2(4x_2 - x_1)}{12566 x_1^3 (x_2 - x_1)} + \frac{2.26}{12566 x_1^2} \le 0$$

$$g_3(x) = 1 - \frac{140.54 x_1}{x_2^2 x_3} \le 0 \qquad g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \le 0$$

where

$$0.05 \le x_1 \le 0.2 \qquad 0.25 \le x_2 \le 1.3 \qquad 2.0 \le x_3 \le 15.0$$

From the Table 7 one can observe that the variants 2, 3 and 4 were able to find the better results for the Tension/Compression Spring design using 1000 generations. The final volume reached is equal to 0.0126973. Using 5000 generations, the previous result for the best volume remains the same (0.0126973) and is again found by the variants 2, 3 and 4 of the APM (Table 8).

Variant	best	median	Mean	std	worst	frun
1	0.0127181	0.0136638	0.0144048	1.82E-03	0.0178499	25
2	**0.0126973**	**0.0135445**	**0.0144046**	1.79E-03	0.0178499	25
3	**0.0126973**	**0.0135445**	0.0144074	1.80E-03	0.0178674	25
4	**0.0126973**	0.0135456	0.0143907	1.78E-03	0.0178499	25
5	0.0127255	0.0140187	0.0146077	1.83E-03	0.0178499	25

Table 7. Results of the coil spring design using 1000 generations

Variant	best	median	mean	std	worst	frun
1	0.0127181	0.0135456	0.0143920	1.82E-03	0.0178499	25
2	**0.0126973**	**0.0135445**	0.0143769	1.78E-03	0.0178499	25
3	**0.0126973**	**0.0135445**	**0.0143706**	1.78E-03	0.0178499	25
4	**0.0126973**	**0.0135445**	0.0143868	1.80E-03	0.0178499	25
5	0.0127255	0.0127255	0.0146097	1.84E-03	0.0178499	25

Table 8. Results of the coil spring design using 5000 generations

5.5 Test 5 - The 10-bar truss design

This well known test problem (Goldberg & Samtani, 1986), which corresponds to the weight minimization of a 10-bar truss, will be analyzed here subject to constraints involving the stress in each member and the displacements at the nodes. The design variables are the cross-sectional areas of the bars (A_i, $i = 1,10$). The allowable stress is limited to $\pm$ 25ksi and the displacements are limited to 2 in, in the x and y directions. The density of the material is 0.1 lb/in^3, Young's modulus is E = 10^4 ksi, and vertical downward loads of 100 kips are applied at nodes 2 and 4.

Two cases are analyzed: discrete and continuous variables. For the discrete case the values of the cross-sectional areas (in^2) are chosen from the set of 42 options: 1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50. Using a six-bit string for each design variable, a total of 64 strings are available which are mapped into the 42 allowable cross sections. For the continuous case the minimum cross sectional area is equal to 0.1 in^2.
For the discrete case all of the variants of the APM were able to find the best result for the final weight of the 10-bar truss structure (5490.738) as detailed in the Table 19. For the continuous case, the original APM (variant 1) found the best solution for the final weight (5062.6605) shown in the Table 10.

variant	best	median	mean	std	worst	frun
1	**5490.7378**	5599.4444	5605.6312	1.29E+02	6156.3977	25
2	**5490.7378**	5634.5138	5647.7405	1.94E+02	6487.3525	25
3	**5490.7378**	**5585.0667**	5636.5571	2.12E+02	6561.2738	25
4	**5490.7378**	5628.1253	5669.3382	2.00E+02	6550.6018	25
5	**5490.7378**	5596.3125	**5600.6728**	6.38E+01	5719.3412	25

Table 9. Results of the 10-bar truss with respect to the discrete case

variant	best	median	mean	std	worst	frun
1	**5062.6605**	**5088.1406**	5116.4183	9.20E+01	5527.0238	25
2	5068.5702	5089.9409	5129.8554	1.20E+03	5631.3084	25
3	5066.8124	5093.6549	**5106.9311**	5.71E+01	5306.6538	25
4	5068.5005	5091.6265	5138.9569	1.49E+02	5774.8811	25
5	5063.6410	5087.2172	5126.4837	1.37E+02	5705.4190	25

Table 10. Results of the 10-bar truss with respect to the continuous case

5.6 Test 6 - The 120-bar truss

This is a three-dimensional dome structure corresponding to a truss with 120 members (Saka & Ulker, 1991, Ebenau et al., 2005, Capriles et al, 2007).
The dome is subject to a downward vertical equipment loading of 600 kN at its crown (node 1). The displacements are limited to 2 cm, in the x, y and z directions. The density of material is 7860 kg/m^3 and Young's modulus is equal to 21000 kN/cm^2. The maximum normal stress is limited by $\sigma_{max} = 32.273\ kN/cm^2$ in tension and compression. Moreover,

the buckling in members under compression is considered, and the stress constraints are given now by:

$$\frac{|\sigma_j|}{\sigma_j^E} - 1 \le 0, \quad i = 1,2,\ldots, p_\sigma^E \quad \text{with} \quad \sigma_j^E = \frac{kEA_j}{L_j^2}$$

where σ_j is the buckling stress at the j-th member, σ_j^E is the maximum allowable buckling stress for this member, and k is the buckling coefficient, adopted equal to 4. The total number of constraints is equal to 351.
The cross-sectional areas are the sizing design variables and they are to be chosen from a Table containing 64 options (see Capriles et al, 2007).

Variant	best	median	mean	std	worst	frun
1	**59.5472**	59.9999	**60.0266**	2.50E-01	60.7749	25
2	**59.5472**	60.0459	60.2679	6.33E-01	62.3707	25
3	**59.7145**	60.0459	60.7749	3.05E+00	75.1747	25
4	**59.5472**	60.0459	60.8720	3.05E+00	75.1747	25
5	**59.5472**	**59.9748**	60.0711	4.03E-01	61.0417	25

Table 11.Results of the 120-bar truss

In this experiment the five variants of the APM were able to find the same best solution (59.5472). The fifth variant of the APM provided the best value of the median (59.9748) and the original APM presented the best result corresponding to the mean value.

6. Conclusion

From the experiments performed here one can see that the original APM found more often the best solutions. Among the new variants, the variants 3 and 5 were the most successful. As the performance of variant 4 was slightly inferior to that of variants 3 and 5, one may suspect that the inclusion of the damping procedure was not advantageous.
A more important conclusion may be that there were not large discrepancies among the results obtained by the original APM and the four new variants, which provide computational savings, and that robust variants can be obtained from the overall family of adaptive penalty methods considered here.

7. References

Barbosa, H.J.C. & Lemonge, A.C.C. (2002). An adaptive penalty scheme in genetic algorithms for constrained optimization problems. *Proceedings of the Genetic and Evolutionary Computation Conference* - GECCO, pp. 287–294, 9-13 July 2002. Morgan Kaufmann Publishers, New York.

Barbosa, H.J.C. & Lemonge, A.C.C. (2003). A new adaptive penalty scheme for genetic algorithms. *Informations Sciences*, Vol. 156, No. 5, pp. 215–251, 2003.

Lemonge, A.C.C. & Barbosa, H.J.C. (2004). An adaptive penalty scheme for genetic algorithms in structural optimization. *Int. Journal for Numerical Methods in Engineering*, Vol. 59, No. 5, pp. 703–736, 2004.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Reading, MA, 1989.

Shoenauer, M. & Michalewicz, Z. (1996). Evolutionary computation at the edge of feasibility. In H-M. Voigt; W. Ebeling; I. Rechenberg and H-P. Schwefel, editors, *Parallel Problem Solving from Nature* - PPSN IV, Vol. 1141, pp. 245–254, Berlin, 1996. Springer-Verlag. LNCS.

Koziel, S. & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, Vo. 7, No. 1, pp. 19–44, 1999.

Liepins, G.E. & Potter, W.D. (1996). A genetic algorithm approach to multiple-fault diagnosis. In L. Davis, editor, *Handbook of Genetic Algorithm*, Chapter 7, pp. 237–250, Boston, USA, 1996. International Thomson Computer Press.

Orvosh, D. & Davis, L. (1994). A genetic algorithm to optimize problems with feasibility constraints. *Proc. Of the First IEEE Conference on Evolutionary Computation*, pp. 548–553. IEEE Press, 1994.

Adeli, H. & Cheng, N.-T. (1994). Augmented Lagrangian Genetic Algorithm for Structural Optimization. *Journal of Aerospace Engineering*, Vol. 7, No. 1, pp. 104–118, January 1994.

Barbosa, H.J.C. (1999) A coevolutionary genetic algorithm for constrained optimization problems. *Proc. of the Congress on Evolutionary Computation*, pp. 1605–1611, Washington, DC, USA, 1999.

Surry, P.D. & Radcliffe, N.J. (1997). The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, Vol. 26, No. 3, 1997.

Runarsson, T.P. & Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 284–294, September 2000.

Kampen, A.H.C. van, Strom, C.S. & Buydens, L.M.C.(1996). Lethalization, penalty and repair functions for constraint handling in the genetic algorithm methodology. *Chemometrics and Intelligent Laboratory Systems*, Vo. 34, pp. 55–68, 1996.

Michalewicz, Z. & Shoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, Vol. 4, No. 1, pp. 1–32, 1996.

Hinterding, R. &. Michalewicz, Z. (1998). Your brains and my beauty: Parent matching for constrained optimization. *Proc. of the Fifth Int. Conf. on Evolutionary Computation*, pp. 810–815, Alaska, May 4-9, 1998.

Koziel, S. & Michalewicz, Z. (1998). A decoder-based evolutionary algorithm for constrained optimization problems, *Proc. of the Fifth Parallel Problem Solving* - PPSN. T. Bäck; A.E. Eiben; M. Shoenauer and H.-P. Schwefel, Editors Amsterdam, September 27-30 1998. Spring Verlag. Lecture Notes in Computer Science.

Kim, J.-H. & Myung, H. (1997). Evolutionary programming techniques for constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, Vol. 2, No. 1, pp. 129–140, 1997.

Le Riche, R.G; Knopf-Lenoir, C. & Haftka, R.T. (1995). A segregated genetic algorithm for constrained structural optimization. *Proc. of the Sixth Int. Conf. on Genetic Algorithms*, pp. 558–565, L.J. Eshelman Editor, Pittsburgh, PA., July 1995.

Homaifar, H.; Lai, S.H.-Y. & Qi, X. (1994). Constrained optimization via genetic algorithms. Simulation, Vol. 62, No. 4, pp. 242–254, 1994.

Joines, J. & Houck, C. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. *Proc. of the First IEEE Int. Conf. on Evolutionary Computation*. Z. Michalewicz; J.D. Schaffer; H.-P. Schwefel; D.B. Fogel and H. Kitano, Editors, pp. 579–584, June 19-23, 1994.

Powell, D. & Skolnick, M.M. (1993). Using genetic algorithms in engineering design optimization with non-linear constraints. *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, In S. Forrest, Editor, pp. 424–430, San Mateo, CA, 1993. Morgan Kaufmann.

Deb, K. (2000) An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, Vol. 186, Nos. 2-4, pp. 311–338, June 2000.

Michalewicz, Z. (1995). A survey of constraint handling techniques in evolutionary computation. Proc. of the 4th Int. Conf. on Evolutionary Programming, pp. 135–155, Cambridge, MA, 1995. MIT Press.

Michalewicz, Z; Dasgupta, D; Le Riche, R.G. & Shoenauer, M. (1996). Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering Journal*, Vol. 30, No. 2, pp. 851–870, 1996.

Bean, J.C. & Alouane, A.B. (1992). A dual genetic algorithm for bounded integer programs. *Technical Report TR 92-53*, Departament of Industrial and Operations Engineering, The University of Michigan, 1992.

Shoenauer, M. & Xanthakis, S. (1993). Constrained GA optimization. *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, S. Forrest, editor, pp. 573–580, Los Altos, CA, 1993. Morgan Kaufmann Publishers.

Coit, D.W.; Smith, A.E. & Tate, D.M. (1996). Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*, Vo. 6, No. 2, pp. 173–182, 1996.

Yokota, T ; Gen, M. ; Ida, K. & Taguchi, T. (1995). Optimal design of system reliability by an improved genetic algorithms. *Trans. Inst. Electron. Inf. Comput. Engrg.*, J78-A(6), pp. 702–709, 1995.

Gen, M. & Gheng, R. (1996a). Interval programming using genetic algorithms. *Proc. of the Sixth International Synposium on Robotics and Manufactoring*, Montepelier, France, 1996.

Gen, M. & Gheng, R. (1996b). A survey of penalty techniques in genetic algorithms. *Proc. of the 1996 International Conference on Evolutionary Computation*, IEEE, pp. 804–809, Nagoya, Japan, UK, 1996.

Hamida, S.B. & Shoenauer, M. (2000). An adaptive algorithm for constrained optimization problems. *Parallel Problem Solving from Nature* - PPSN VI, Vo. 1917, pp. 529–538, Berlin. Springer-Verlag. Lecture Notes in Computer Science, 2000.

Zavislak, M.J. (2004). Constraint handling in groundwater remediation design with genetic algorithms, 2004. *M.Sc. Thesis*, Environmental and Civil Engineering, The Graduate College of the Illinois University at Urbana-Champaign.

Gallet, C. ; Salaun, M. & Bouchet, E. (2005). An example of global structural optimisation with genetic algorithms in the aerospace field. *VIII International Conference on Computational Plasticity* - COMPLAS VIII, CIMNE, Barcelona, 2005.

Obadage, A.S. & Hampornchai, N. (2006). Determination of point of maximum likelihood in failure domain using genetic algorithms. *International Journal of Pressure Vessels and Piping*, Vol. 83, No. 4, pp. 276–282, 2006.

Sandgren, E. (1998). Nonlinear integer and discrete programming in mechanical design. *Proc. of the ASME Design Technology Conference*, pp. 95–105, Kissimee, FL, 1988.

Kannan, B.K. & Kramer, S.N. (1995). An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design*, Vol. 116, pp. 405–411, 1995.

Deb, K. (1997). GeneAS: A robust optimal design technique for mechanical component design. *Evolutionary Algorithms in Engineering Applications*, pp. 497–514. Springer-Verlarg, Berlin, 1997.

Coello, C.A.C. (2000). Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems. *Computers in Industry*, Vol. 41, No. 2, pp. 113–127, January 2000.

Erbatur, F. ; Hasançebi, I. ; Tütüncü, I. & Kilç, H. (2000). Optimal design of planar and space structures with genetic algorithms. *Computers & Structures*, Vo. 75, pp. 209–224, 2000.

Goldberg, D.E. & Samtani, M.P. (1986). Engineering optimization via genetic algorithms. *Proc. 9th Conf. Electronic Computation*, ASCE, pp. 471–482, New York, NY, 1986.

Saka, M.P. & Ulker, M. (1991). Optimum design of geometrically non-linear space trusses. *Computers and Structures*, Vol. 41, pp. 1387–1396, 1991.

Ebenau, C. ; Rotsschafer, J. & Thierauf, G. (2005). An advance evolutionary strategy with an adaptive penalty function for mixed-discrete structural optimisation. *Advances in Engineering Software*, Vo. 36, pp. 29–38, 2005.

Capriles, P.V.S.Z.; Fonseca, L.G. ; Barbosa, H.J.C & Lemonge, A.C.C. (2007). Rank-based ant colony algorithms for truss weight minimization with discrete variables. *Communications in Numerical Methods in Engineering*, Vol. 26, No. 6, pp. 553–576, 2007.