A Genetic Algorithm Based Augmented Lagrangian Method for Accurate, Fast and Reliable Constrained Optimization

Kalyanmoy Deb[*] and Soumil Srivastava
Kanpur Genetic Algorithms Laboratory (KanGAL)
Indian Institute of Technology Kanpur
Kanpur, UP, India, PIN 208016
deb@iitk.ac.in, soumil.iitk@gmail.com
http://www.iitk.ac.in/kangal/pub.htm

KanGAL Report Number 2011012

Abstract

Among the penalty based approaches for constrained optimization, augmented Lagrangian (AL) methods are better in at least three ways: (i) they have theoretical convergence properties, (ii) they distort the original objective function minimally, thereby providing a better function landscape for search, and (iii) they can result in computing optimal Lagrange multiplier for each constraint as a by-product. Instead of keeping a constant penalty parameter throughout the optimization process, these algorithms update the parameters (called multipliers) adaptively so that the corresponding penalized function dynamically changes its optimum from the unconstrained minimum point to the constrained minimum point with iterations. However, the flip side of these algorithms is that the overall algorithm requires a serial application of a number of unconstrained optimization tasks, a process that is usually time-consuming and tend to be computationally expensive. In this paper, we devise a genetic algorithm based parameter update strategy to a particular AL method. The proposed strategy updates critical parameters in a self-adaptive manner based on population statistics. Occasionally, a classical optimization method is used to improve the GA-obtained solution, thereby providing the resulting hybrid procedure its theoretical convergence property. The GAAL method is applied to a number of constrained test problems taken from the evolutionary algorithms (EAs) literature. The number of function evaluations required by GAAL in most problems is found to be smaller than that needed by a number of existing evolutionary based constrained handling methods. GAAL method is found to be accurate, computationally fast, and reliable over multiple runs. Besides solving the problems, the proposed GAAL method is also able to find the optimal Lagrange multiplier associated with each constraint for all test problems as an added benefit – a matter that is important for a sensitivity analysis of the obtained solution, but has not yet been paid adequate attention in the past evolutionary constrained optimization studies.

1 Introduction

Constrained optimization problems are the most important and ubiquitous type of engineering optimization problems. Evolutionary algorithms (EA) have been applied extensively for tackling these problems with various degrees of success. The penalty function approach is a relatively simple and remarkably easy to implement and therefore has been popularly used with an EA. But

[*]K. Deb is also an adjunct professor at the Department of Information and Service Economy, Aalto University, Helsinki, Finland.

the approach also has several functional and operational drawbacks which limit its precision and speed of convergence [6, 19]. First, the penalty function approaches do not take into account the mathematical form (linear or non-linear, convex or non-convex, unimodal or multi-modal, etc.) of the constraints. Second, the addition of the penalty term on the objective function usually makes a distortion in the objective function, which may create artificial non-linearity and multi-modalities that may thwart the progress of an optimization algorithm to the true constrained minimum point. Third, most penalty function methods do not have a convergence proof and works under the assumption that optimal solutions of the subsequent penalized functions approach the true constrained minimum of the problem. Various types of sophisticated penalty function methods have been devised in the past to overcome some of these drawbacks in the context of evolutionary algorithms, such as multi-level penalty functions [11], dynamic penalty functions [12], and others. But according to a study by Michalewicz and Schoenauer [16], such sophisticated methods are less generic and work well on some problems but not on others.

In 2000, the first author had proposed a parameter-less approach [7] for constrained handling within a GA that eliminated the need for a penalty parameter and was received quite favorably. Other popular approaches used a two-objective method [24] or, in general, a multi-objective approach [4] for handling constraints. Other approaches used other evolutionary algorithms, such as the particle swarm optimization (PSO) [25], differential evolution (DE) [21] and reported excellent performance. Most of these studies in evolutionary algorithms that used a penalized technique for handling infeasible solutions have used some form of a penalty function. However, the augmented Lagrangian methods have received a large attention in the recent past for solving constrained optimization problems [1, 2, 17, 23, 26]. The augmented Lagrangian methods are better than direct penalty function approaches in a number of ways:

1. AL methods usually have a theoretical convergence property to a Karush-Kuhn-Tucker (KKT) point. The particular AL approach used in this study has such a property.

2. The AL approach does not distort the original objective function, instead it has an effect of *shifting* the objective function in a manner so that its optimum moves from the unconstrained minimum to the constrained minimum point. This does not change the complexity of the problem too much from start to finish and makes it easier to choose a single optimization algorithm for the entire problem-solving task.

3. As a by-product, AL methods find the Lagrange multiplier value for each constraint. Lagrange multipliers are important for subsequent post-optimality analysis and for sensitivity analysis [18, 19, 13]. The ability to compute Lagrange multipliers provides AL methods a definite edge over penalty function approaches and for that matter over most other optimization algorithms.

However, one of the criticisms of AL methods is the need for solving a number of serial optimization tasks, which eventually makes the overall approach computationally expensive. Also, a current optimization task depends largely on the accuracy of previous optimization tasks, thereby requiring to solve every optimization problem with an appropriate accuracy. Another matter which is not adequately discussed in the literature, but is also an important factor is that particularly in solving multi-modal constrained problems, if an earlier optimization task did not converge to an appropriate optimum point that would eventually lead the process to the true constrained optimum, the value of the associated parameters computed at the *false* optimum will be different from that are required for the algorithm to converge to the true constrained minimum point. The use of a more global optimization algorithm to solve every intermediate problem may alleviate such a false convergence.

In this paper, we revisit the use of a classical augmented Lagrangian (AL) approach, but instead of using a classical unconstrained optimization algorithm to solve the resulting penalized functions one at a time, we extend our previous pilot study [20] here and employ a genetic algorithm to seamlessly solve penalized functions by updating the associated parameters adaptively. For each constraint (of equality and inequality type), the proposed AL method involves a dynamically changed parameter. The population approach of the proposed GA-based method maintains diversity of solutions and allows a more global search of the individual optimization tasks. Furthermore, the switching from one parameter setting to another is made completely in a self-adaptive manner based on population statistics so that once started the user does not have to control the switching from one optimization task to another, as it is the case in classical way of applying the AL method. However, to induce convergence property of the proposed algorithm, a classical optimization procedure (the `fmincon` routine of MATLAB) is embedded within the proposed GA-based procedure to terminate a run.

In the remainder of the paper, we first describe the proposed AL-based constraint handling method and the various operators (such as the adaptive mutation and self-adaptive penalty parameter update procedure) incorporated within it to ensure a fast and accurate convergence. We then recommend values for the parameters based on two parametric studies. Thereafter, we present the performance of the proposed method on a number of standard constrained test problems taken from the EA literature, compare its results to the classical method, and highlight the benefits of the proposed approach. Conclusions of the study are presented next.

2 Augmented Lagrangian Method and the Proposed Algorithm

A general notation for a constrained optimization problem is:

$$\begin{array}{rll} \text{Minimize} & f(\mathbf{x}), & \\ \text{subject to} & g_j(\mathbf{x}) \geq 0, & \forall\, j = 1,2,\ldots,J, \\ & h_k(\mathbf{x}) = 0, & \forall\, k = 1,2,\ldots,K, \\ & x_i^l \leq x_i \leq x_i^u, & \forall\, i = 1,2,\ldots,n. \end{array} \tag{1}$$

In a particular classical augmented Lagrangian method for constrained optimization, the objective function $f(\mathbf{x})$ is modified as follows:

$$\begin{aligned} P(\mathbf{x},\sigma^t,\tau^t) = f(\mathbf{x}) &+ R\sum_{j=1}^{J}\left[\left(\langle g_j(x)+\sigma_j^t\rangle\right)^2 - \left(\sigma_j^t\right)^2\right] \\ &+ R\sum_{k=1}^{K}\left[\left(h_k(x)+\tau_k^t\right)^2 - \left(\tau_k^t\right)^2\right], \end{aligned} \tag{2}$$

where $\langle g(x)\rangle = g(\mathbf{x})$ if $g(\mathbf{x}) < 0$ else it is zero. The superscript t indicates the iteration counter. In solving the t-th optimization problem, multipliers σ_j^t and τ_k^t are kept fixed. The initial values of these multipliers are set to zero, or $\sigma_j^0 = 0$ and $\tau_k^0 = 0$. However, these multipliers are updated at the end of t-th optimization run and are kept fixed for the $(t+1)$-th iteration, as follows:

$$\sigma_j^{t+1} = \langle \sigma_j^t + g_j(\mathbf{x}^t)\rangle, \quad j = 1,2,\ldots,J, \tag{3}$$

$$\tau_k^{t+1} = \tau_k^t + h_k(\mathbf{x}^t), \quad k = 1,2,\ldots,K, \tag{4}$$

where $\mathbf{x}^t$ is the optimum solution of the t-th penalized function P. It is also important to mention that the starting solution of the initial optimization task can be chosen as any arbitrary solution, whereas the initial solution to the t-th $(t \geq 1)$ optimization problem is the optimum solution found at $(t-1)$-th optimization task. This sequential optimization procedure continues till an insignificant change in the optimum solutions of the two consecutive optimization tasks is observed. It is also important to mention that the penalty parameter R is usually kept constant throughout the entire optimization process.

A major result which can be obtained from a KKT analysis of the above penalized optimization problem is that at a theoretical termination of the algorithm (say, occurring at iteration T), the optimal Lagrangian multipliers for the original problem can be calculated by using the following relationships:

$$u_j = -2R\sigma_j^T, \tag{5}$$

$$v_k = -2R\tau_k^T. \tag{6}$$

Although the above description and its associated theoretical result are attractive, there are some practical implementational issues which we discuss next. First, since the value of R is kept fixed throughout the optimization process, it may be apparent that its value can be fixed to any arbitrary value. Although theoretically it may be justified, in practice the performance of an algorithm largely depends on the chosen value of R. Notice that all constraints are multiplied by the same penalty parameter R. It is discussed enough in the context of penalty parameter approaches that this causes some constraints to behave as more important than others. Due to the possibility of constraints taking values of dissimilar orders of magnitude, unequal treatment of constraints may not allow the overall algorithm to achieve a systematic convergence to the constrained minimum. Constraints taking higher magnitude of values will get emphasized more. Thus, there is a need for normalizing the constraint functions before using them in Equation 2 [6, 10]. Second, the value of R makes a balance between the objective function f and the constraint violation given by the second and third terms of the right side of Equation 2. Unless a proper balance is established, either the objective function is not emphasized enough or the constraint violation is not adequately emphasized. The above scenario will result in a premature convergence to an arbitrary feasible solution or to an infeasible solution, respectively. Importantly, for arriving at a computationally efficient approach, such a balance must be made adaptively with more emphasis to constraint violation in the beginning of the optimization process and then a relatively larger emphasis to the objective function towards later generations. As done in penalty function approaches, in AL method as well, there is a need for updating R with the optimization iteration counter t. We have proposed here a self-adaptive approach here for updating R by using a statistics of the current population objective values. Once a new R is set at the end of some iterations, the corresponding multipliers are also updated by using the theoretical relationships presented in Equations 5 and 6. Since optimal Lagrange multipliers u_j and v_k are properties of the supplied optimization problem, the equations suggest that the new σ_j and τ_k values must be changed in an inversely proportionate manner with the new R value.

Based on these discussions, we suggest a GA based AL method with the following properties:

1. GAAL self-adaptively alters the value of R whenever a sufficient convergence is observed.

2. GAAL self-adaptively determines the update of multiplier values σ_j from one iteration to another and based on a theoretical basis whenever there is a change in R value.

3. GAAL uses a termination condition based on solutions obtained for successive R updates and its subsequent classical optimization run.

4. GAAL is seamless in that no user intervention is needed once the algorithm is started.

We now discuss the details of the proposed GAAL method.

2.1 Proposed GAAL Method

In this study, we restrict ourselves to inequality constraints only, however extensions of the current study can be made for equality constraints as well. In the following, we first present the proposed algorithm and then discuss its individual operators:

1. **Initialization**
   (a) Set $t = 0$. Initialize the GA population $\mathcal{P}^t$ with N members randomly.
   (b) Set $\sigma_j^t = 0$ for all constraints, $j = 1, 2, \ldots, J$.
   (c) Set the initial penalty parameter R based on the statistics of the initial population as follows:
   $$R = 50 \frac{\sum_N |f(\mathbf{x})|}{\sum_N \mathrm{CV}(\mathbf{x})}, \tag{7}$$
   where $\mathrm{CV}(\mathbf{x}) = \sum_{j=1}^{J} \langle \hat{g}_j(x) \rangle^2$ and $\hat{g}_j$ is the normalized version of the constraint function g_j. Set $R_{\text{old}} = R$.
   (d) Record the best population member $\mathbf{x}^{b,t}$ and increment t by one.

2. For every generation t, perform the following steps:
   (a) Perform one generation of the GA using the binary tournament selection operator with the penalized function $P()$ (equation 2) as fitness measure to be minimized, the simulated binary crossover (SBX) operator [8], and the adaptive polynomial mutation operator [15] on population $\mathcal{P}^t$ to create a new offspring population $\mathcal{Q}^t$ of size N.
   (b) Choose the best population member $\mathbf{x}^{b,t}$ of $\mathcal{Q}^t$. Set $\mathcal{P}^t = \mathcal{Q}^t$ and compute $\rho = \left| \frac{P(\mathbf{x}^{b,t}, \sigma^t) - P(\mathbf{x}^{b,t-1}, \sigma^{t-1})}{P(\mathbf{x}^{b,t-1}, \sigma^{t-1})} \right|$.
   (c) If $(t \bmod \kappa) = 0$ and $\rho \leq \delta_1$,

      Reset penalty parameter R as follows:
      $$R = \omega \frac{10 \sum_N |f(\mathbf{x})|}{N} + (1 - \omega) R_{\text{old}}, \tag{8}$$
      and update multipliers as follows:
      $$\sigma_j^t = \sigma_j^t \frac{R_{\text{old}}}{R}, \quad j = 1, 2, \ldots, J.$$
      Set $R_{\text{old}} = R$ and employ a classical optimization algorithm with the current best population member $(\mathbf{x}^{b,t})$ as the initial solution to find a new solution $\mathbf{y}^t$. Replace the worst solution of the population $\mathcal{P}^t$ by $\mathbf{y}^t$. Increment counter t by one and go to Step 3.

   Else go to Step 2(d).
   (d) If $\rho \leq \delta_2$, update the multipliers using:
   $$\sigma_j^{t+1} = \langle \sigma_j^t + g_j(\mathbf{x}^{b,t}) \rangle, \quad j = 1, 2, \ldots, J.$$
   Increment counter t by one and go to Step 2(a).

3. **Termination check:** If the absolute difference between $f(\mathbf{y}^t)$ and $f(\mathbf{y}^{t-1})$ is less than or equal to a pre-specified small number ϵ, report $\mathbf{y}^t$ as the GAAL optimized solution, compute Lagrange multipliers using Equation 5, and the procedure is terminated. Otherwise, increment counter t by one and go to Step 2(a).

The automatic update of penalty parameter is an important feature of this algorithm. Initially, the penalty parameter R is set such that the average constraint violation penalty is scaled to 50 (chosen after some trial and error experiments) times the average absolute objective function value. This provides the algorithm adequate flexibility to venture into both infeasible and feasible regions while maintaining moderately high penalty on infeasible regions. Subsequently, during the course of the run, the penalty parameter is updated according to Equation 8 as and when the rate of improvement in penalized function value is below a certain threshold. The weighted transition approach with a weight ω to the new calculated parameter allows a gradual change in R. This facilitates in a smoother convergence of the algorithm, as we shall demonstrate later.

Importantly, we update the penalty parameter R in Step 2(c) and the multipliers σ_j^t in Step 2(d) at two different levels in this method. If the improvement in the best population member ($\mathbf{x}^{b,t}$) is less than δ_2 (a user-defined parameter) then we update the multipliers alone and continue. If this update does not yield better results in the succeeding generations below another threshold (δ_1), we make a change in the value of R for the algorithm to look for better solutions elsewhere in the search space. Parameters δ_1 and δ_2 must be set such that $\delta_2 \gg \delta_1$. This is to ensure that the multiplier update (Step 2(d)) occurs more often than the R update (Step 2(c)). From the relationship between u_j, σ_j and R given in Equation 5, we then update the multipliers in a way so as to make the product of R and σ_j constant.

The SBX operator is the same as that described in [8]. However, we use an adaptive mutation operator in which the mutation probability is changed with the generation counter as follows [9]:

$$p_m = \frac{0.1}{n} + \frac{0.1t}{t_{max}}\left(1 - \frac{1}{n}\right), \tag{9}$$

where n is the number of variables, t is the number of generations and t_{max} is the maximum number of generations set by the user during the run. This creates about 1% perturbance in the solutions in the first generation and subsequently decreases the perturbance in later generations. The constant values of 50 and 10 used in Equations 7 and 8, respectively, are found to produce better results through trial-and-error experiments. Other parameters used in the algorithm are set as follows:

1. Population size N: We recommend the use of $N = \max(10n, 50)$, where n is the number of variables in the problem.

2. Probability of crossover p_c and SBX index η_c: We recommend $p_c = 0.9$ and $\eta_c = 2$. They are found to work well on all our test problems.

3. Mutation index η_m: We recommend $\eta_m = 100 + t$.

4. Update threshold δ_1: Based on simulation studies, we recommend using $\delta_1 = 0.0001$ or 0.01%. This parameter is used to decide on an update of R and start a classical optimization method as a local search operator.

5. Update threshold δ_2: Based on simulation studies, we recommend using $\delta_2 = 0.1$ or 10%. This parameter is used to decide on an update of the multipliers.

6. R update parameter κ: This parameter is used to prevent too frequent changes in R.

7. Weight parameter ω: This parameter determines the extent of transition towards the newly determined penalty parameter. This is important as it directly affects the speed and nature of convergence.

8. Terminating parameter ϵ: We choose $\epsilon = 0.0001$.

After a solution with four decimal places of accuracy (with $\epsilon = 0.0001$) in the objective value is achieved, the optimized variable vector is presented here with six decimal places.

Next, we perform a parametric study to examine the effect of two important parameters κ and ω. Other parameter values are fixed at the standard values mentioned above or are not found to make any significant change in the performance of our proposed algorithm.

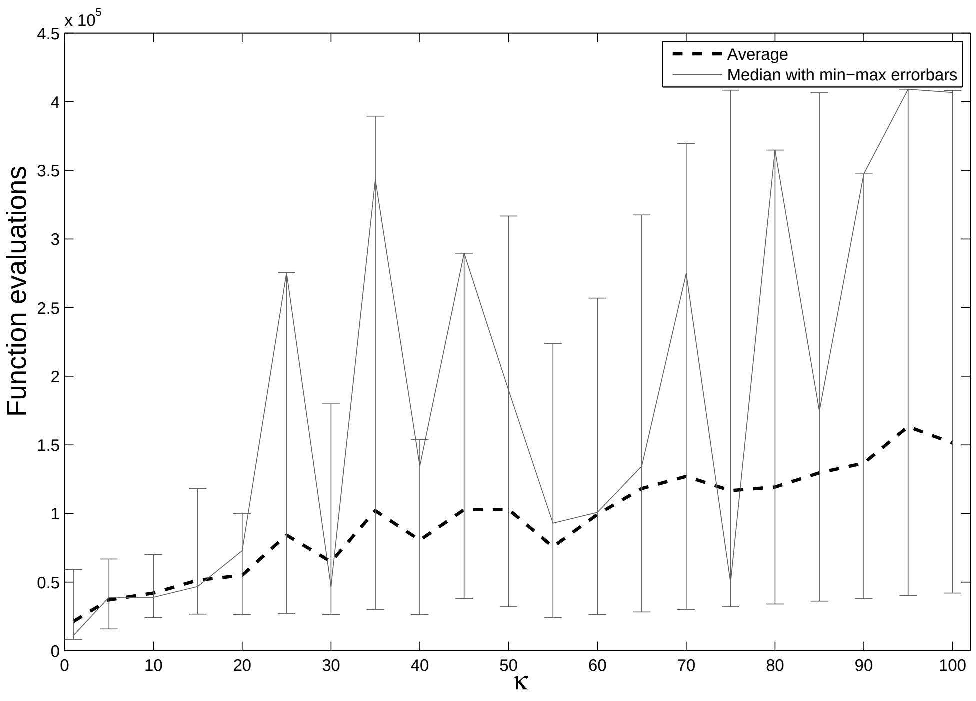

Figure 1: Function evaluations versus κ for g02. As κ is increased, more function evaluations are needed.

2.2 Parametric Study : κ

We performed a parametric study of κ on three constrained optimization problems: g02, g08 and g10, borrowed from the literature [14]. Figures 1, 2 and 3 present the results in a comprehensible manner. It is observed that on increasing the value of κ, more number of function evaluations are required in solving the problems. But, it is also observed that if κ is set to one, that is, allowing `fmincon` routine to be invoked at every generation, the procedure leads to a premature convergence for some problems (g08, for example). Besides, by performing the local search so frequently does not allow the GA procedure enough time to improve the solutions between two local search procedures. Based on the stated observations, we recommend $\kappa = 5$.

2.3 Parametric Study : ω

Next, we perform a parametric study of ω on the same three problems: g02, g08 and g10. The parameter ω is varied within $[0, 1]$. Figures 4, 5 and 6 present the results for the three problems, respectively. It can be observed that the required function evaluations are smaller for intermediate values of ω. Based on the stated observations, we recommend $\omega = 0.5$, in which an equal weight ($\omega = 0.5$) is assigned to both newly calculated R and the previous R. As depicted in the figures, too much emphasis on the old value or the new value does not yield good results.

3 Application on Test Problems

By setting $\kappa = 5$ and $\omega = 0.5$, we now apply our proposed GAAL algorithm to a number of constrained test problems and compare the performance of GAAL with existing evolutionary based constrained handling algorithms.

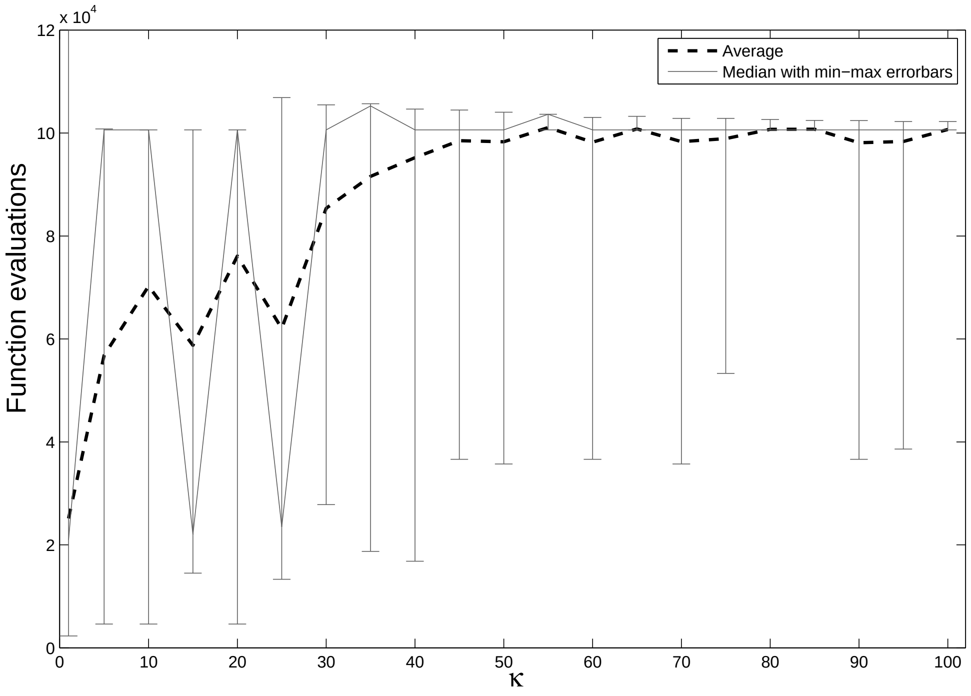

Figure 2: Function evaluations versus κ for g08. As κ is increased, more function evaluations are needed but reaches a plateau after a value of around 50.

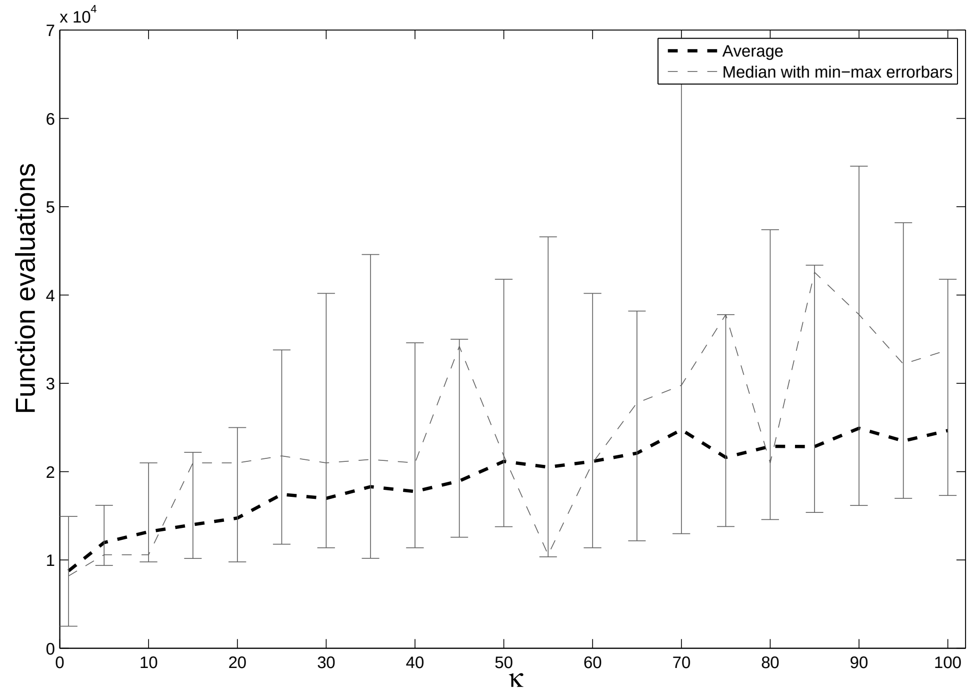

Figure 3: Function evaluations versus κ for g10. As κ is increased, more function evaluations are needed.

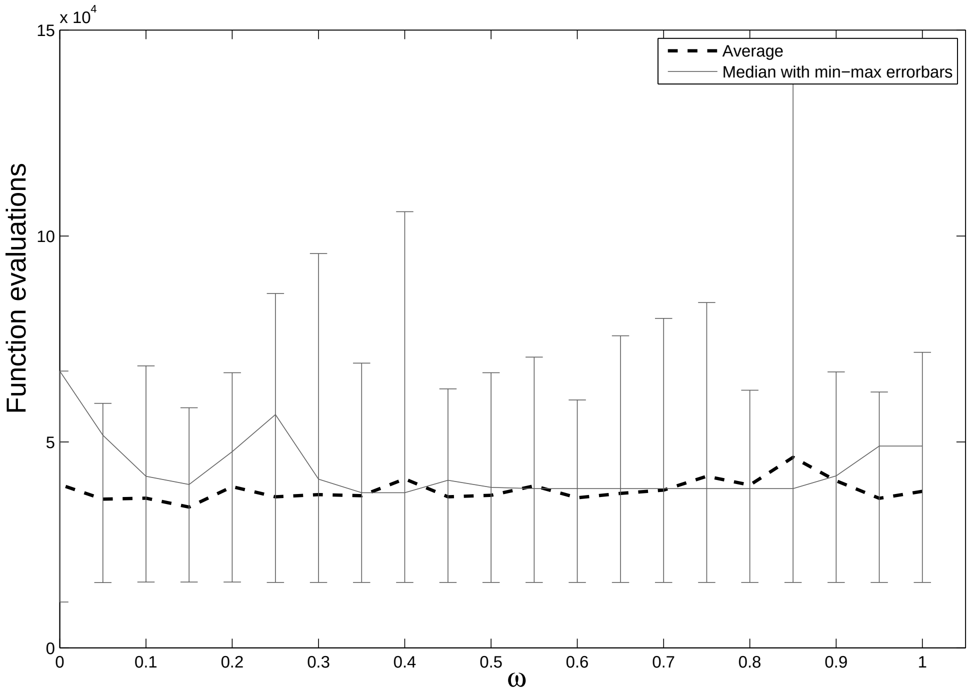

Figure 4: Function evaluations versus ω for g02. ω has little effect on function evaluations for this problem.

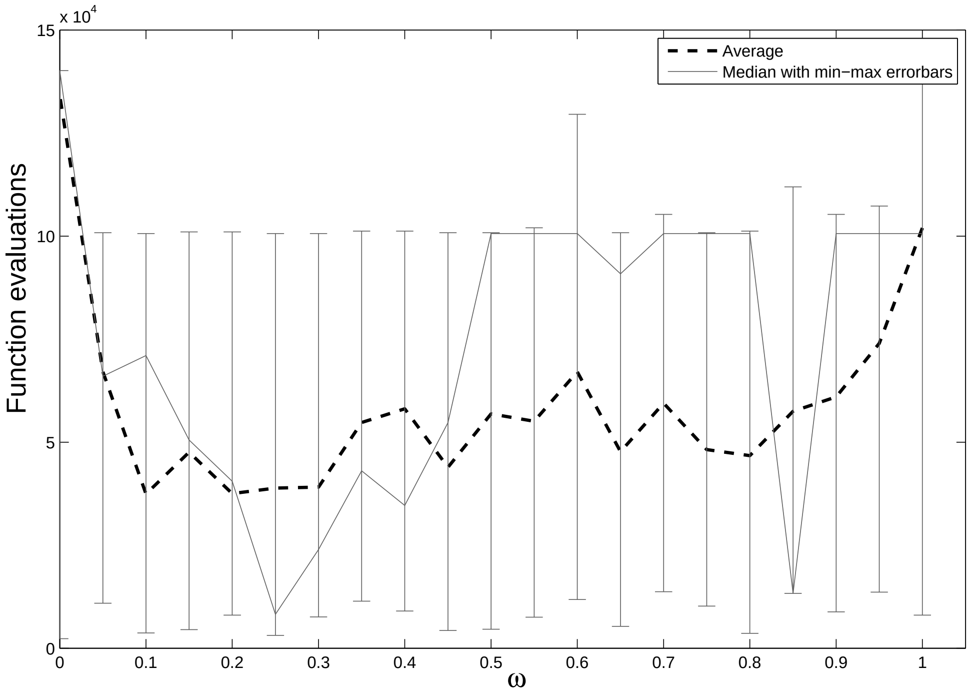

Figure 5: Function evaluations versus ω for g08. As ω is increased beyond 0.8 or decreased below 0.1, more function evaluations are needed.

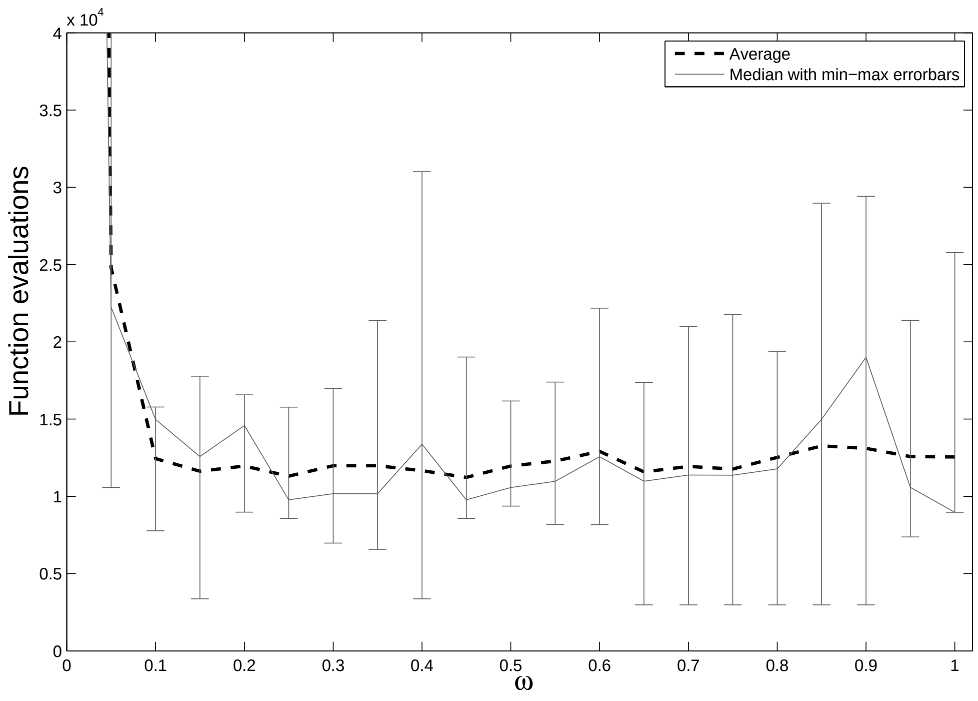

Figure 6: Function evaluations versus ω for g10. As ω is decreased beyond 0.1, required number of function evaluations increase rapidly.

3.1 Problem A: g18

This problem has nine variables and 13 inequality constraints. Refer to appendix for a detail description of this problem. The best known optimum is as follows:

$$\begin{aligned}\mathbf{x}^* =&(-0.657776, -0.153419, 0.323414, -0.946258, -0.657776,\\ &-0.753213, 0.323414, -0.346463, 0.599795),\end{aligned}$$

with $f^* = -0.866025$. The optimal solution obtained by GAAL is shown below:

$$\begin{aligned}\mathbf{x}^{\mathrm{GAAL}} =&(-0.490498, 0.871442, -0.999940, 0.010937, -0.490498,\\ &0.871442, -0.999940, 0.010937, 0.000000),\end{aligned}$$

with $f^{\mathrm{GAAL}} = -0.866025$. Clearly, although there is a difference in the decision variable vectors, both solutions are feasible and the objective function values are identical at least to six decimal places of accuracy. This indicates that the problem is multi-modal having a number of feasible constrained optimal solutions and the best-reported solution lies on one optimum and our GAAL-obtained solution lies on another optimum. The best, median and worst function values obtained in 25 different runs started from different initial populations are -0.866025, -0.866025 and -0.674981, respectively. Only one out of the 25 runs did not find the the optimum accurately. The rest 24 runs of GAAL are within 0.01% of the optimum. The corresponding Lagrangian multipliers for the above-mentioned GAAL solution are as follows: (0.14291952, 0, 0.13948934, 0.14266385, 0, 0.14418376, 0.14966718, 0, 0.14274691, 0, 1.13003191, 0, 0), thereby indicating that constraints g_1, g_3, g_4, g_6, g_7, g_9, and g_{11} are active at the optimum.

We choose this problem to demonstrate the self-adaptive behavior of the associate algorithmic parameters such as the penalty parameter R and multipliers σ_j with the generation counter.

Figure 7 shows the variation of population-best penalized function value P with the generation counter. It is clear from the figure that after some initial fluctuations, the penalized function value stabilizes to its minimum value. A similar trend happens for the corresponding objective function value f (Figure 8). In the initial few generations, the population-best solutions are infeasible, hence the objective values are apparently smaller than $f^* = -0.866025$, but when GAAL manages to find the feasible region, the corresponding objective value is higher. Eventually, the algorithm is able to converge to an optimum solution having $f^* = -0.866025$.

The variations of P and f shown above are typical for penalty-based constrained handling algorithms. Let us now investigate how the algorithmic parameters (such as R and σ_j) self-adapt themselves with the generation counter so as to follow the established theoretical relationship leading to the optimal Lagrange multiplier value for each constraint at the end. First, we plot the variation of the penalty parameter R in Figure 9. Although R can get updated after every $\kappa = 5$ generations, the $\rho \leq \delta_1$ condition does not allow the algorithm to have too frequent changes. At generation 35, the penalty parameter R is increased to a value of 776.25 from 33.27. As Figure 8 reveals, the reason for this drastic change in R is due to a sharp increase in objective values (refer to the R-update equation). Since an appropriate value of R is expected to make a balance between the constraint violation value and the objective function value, our carefully designed R-update rule (Equation 8) enables an increase in R so as to match the range of constraint values with that of the objective values. Thereafter, despite some slight changes in R with further generations, a self-adaptive drastic change of R at generation 35 shifted the focus of the algorithm from infeasible region to feasible region and was an important and necessary event for the success of the algorithm in getting to the constrained minimum solution.

Next, we show the variation of multipliers for a few constraints for the population-best solution with the generation counter in Figures 10 to 15. Starting with with $\sigma_j^0 = 0$, all σ_j^t values reduce as the population-best solution remains infeasible during the initial generations. At generation 35, the penalty parameter R is self-adapted to a high value, thereby making the absolute value of all σ_j values small (recall the inverse relationship between R and σ_j values). Thereafter, the

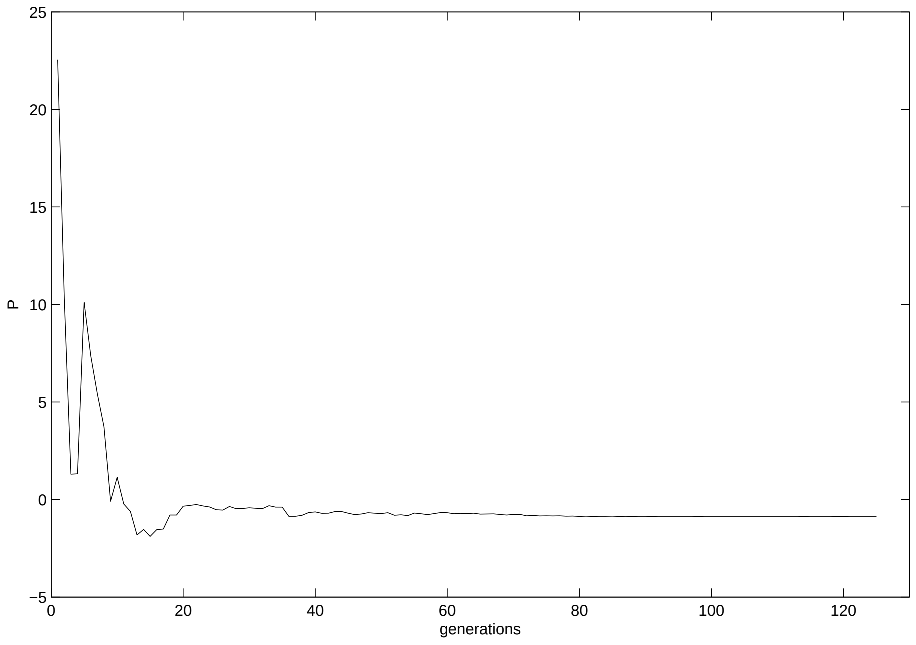

Figure 7: Problem g18: P versus generation counter.

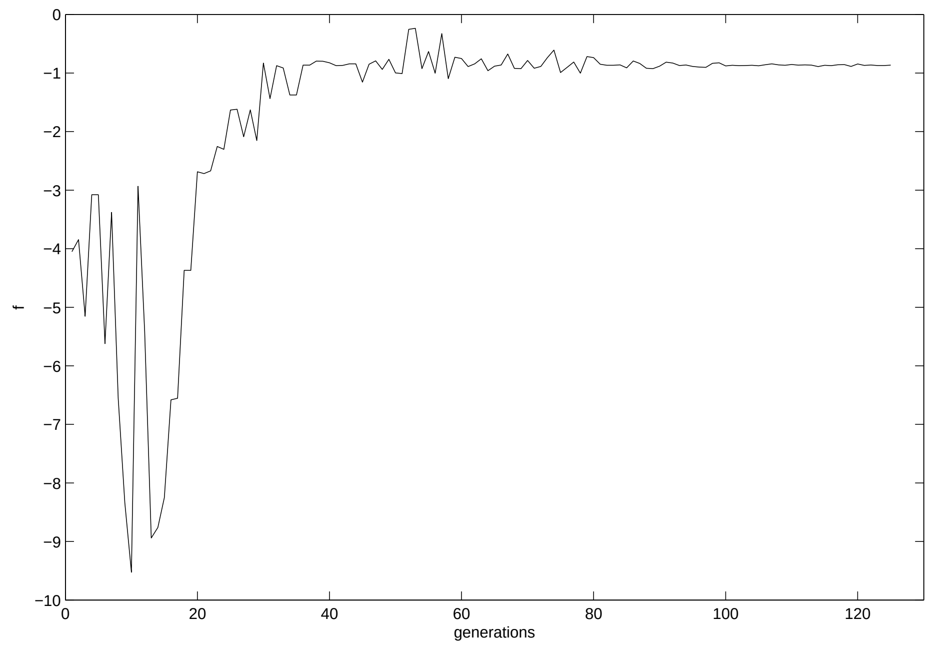

Figure 8: Problem g18: f versus generation counter.

multipliers gets adjusted by the AL rules given in Equation 3. Interestingly, σ_5 returns to a value zero after being negative in the initial generations and make a correct computation of a zero value

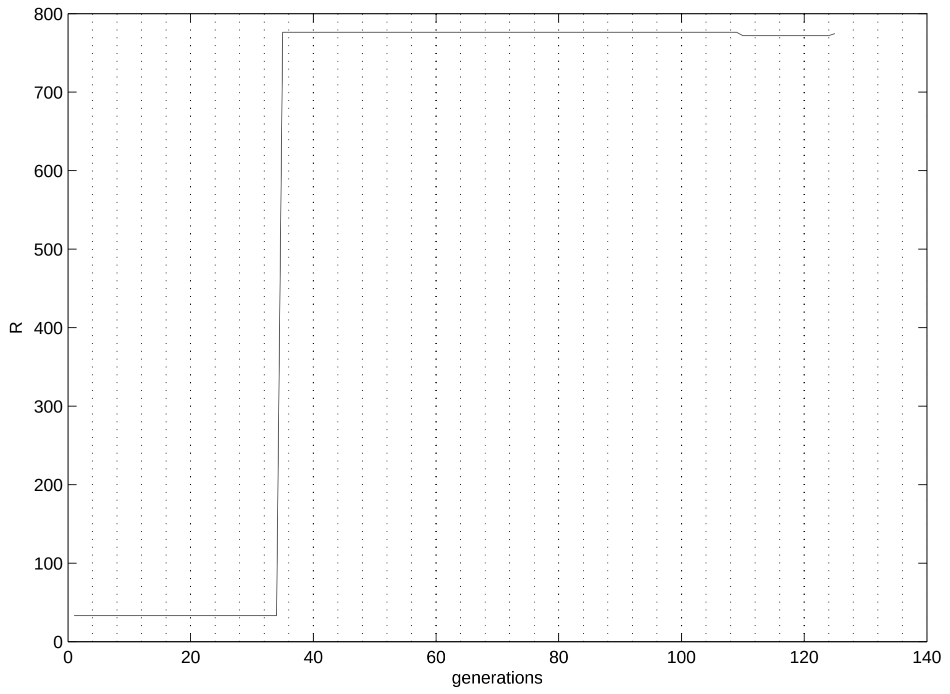

Figure 9: Problem g18: R versus generation counter.

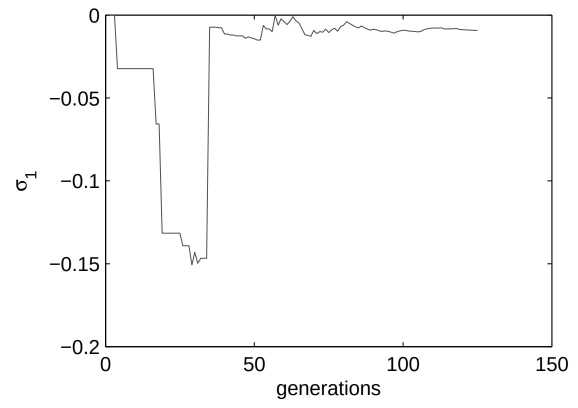

Figure 10: Problem g18: σ_1 versus generation counter.

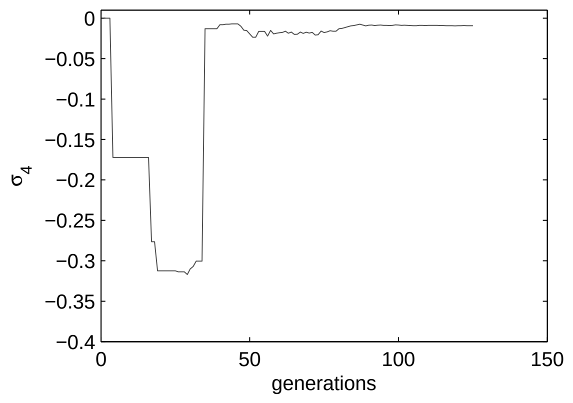

Figure 11: Problem g18: σ_4 versus generation counter.

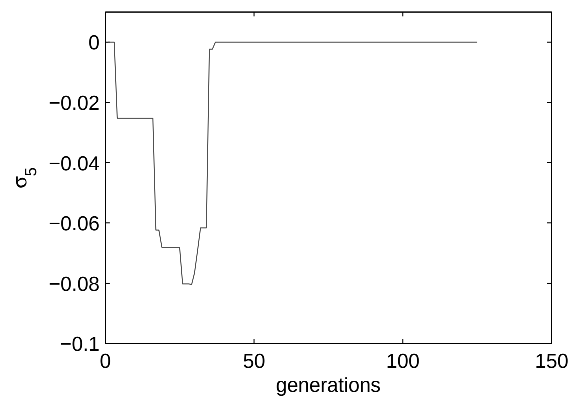

Figure 12: Problem g18 : σ_5 versus generation counter.

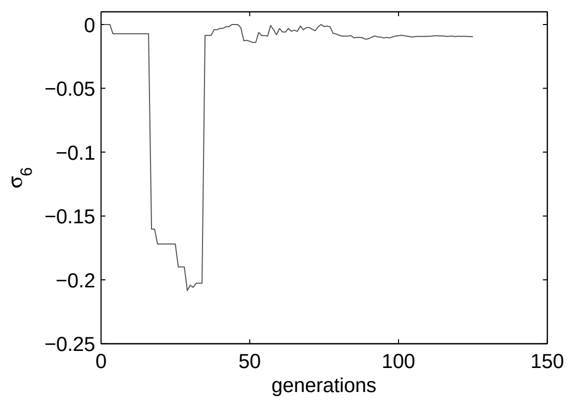

Figure 13: Problem g18: σ_6 versus generation counter.

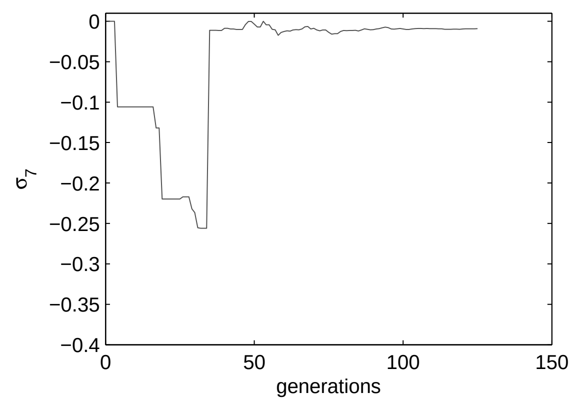

Figure 14: Problem g18: σ_7 versus generation counter.

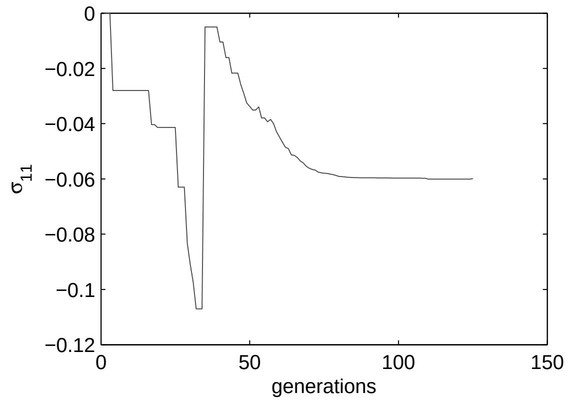

Figure 15: Problem g18: σ_{11} versus generation counter.

of the corresponding Lagrange multiplier value (u_5). Besides the drastic change at generation 35 (mainly due to the change in R at this generation), subsequent changes in multiplier values can also be observed, particularly for σ_{11} in Figure 15. Since all Lagrange multipliers (u_j) must be non-negative at the constrained optimum solution, it is interesting to note that all multiplier values (σ_j) settle down to non-positive values.

The performance of GAAL in this problem amply demonstrates its self-adaptive property and its ability to adjust all parameters associated with 13 constraints to reach near a constrained optimum solution. The occasional updates of R to make a right balance between objective function and the constraint values help steer the search in the right direction. Thereafter, frequent updates of multipliers (σ_j) using the AL principle provided local improvements in the solutions. Importantly, a seamless operation orchestrating the adjustments of multiple parameters in the right direction is achieved by the use of theory of augmented Lagrangian approach and by the global search power of genetic algorithms.

3.2 Problem B: g01

The next problem has 13 variables and nine inequality constraints. The appendix presents the mathematical formulation of the problem. The best-known optimum for this problem is as follows:

$$\mathbf{x}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1).$$

The corresponding $f^* = -15$. Six constraints (all except g_4, g_5 and g_6) are active at this optimum. The best optimized solution of GAAL is given below:

$$\mathbf{x}^{\mathrm{GAAL}} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1),$$

with $f^{\mathrm{GAAL}} = -15$. This solution is identical to best-known solution. The best, median and worst objective values obtained for 25 different runs are -15, -15 and -12.453125, respectively. The proposed method has performed quite well in this problem as well. Only one out of the 25 runs is not able to find the the optimum accurately. The rest 24 runs of GAAL were within 0.01% of the optimum. The corresponding Lagrangian multipliers for the best optimized solution are as follows: $\mathbf{u} = (0.0042346,\ 0.0046562,\ 0.0054315,\ 0,\ 0,\ 0,\ 1.021383,\ 1.012629,\ 0.990493)$, indicating that constraints g_4, g_5, and g_6 are inactive. This agrees with the known result on this problem. For brevity, we do not show any figure for this or other problems demonstrating the variation of the algorithmic parameters with the generation counter.

It is clear from the two applications that a highlight of the use of an AL method in a GA is that in addition to the optimized solution, the proposed GAAL method is also able to supply us with the optimal Lagrange multiplier values which can indicate the relative importance of each constraint to the obtained optimized solution – a matter which is important from a practical standpoint.

3.3 Problem C: g02

This problem has 20 variables and two inequality constraints. The best-known optimum (upto nine decimal places of accuracy) for this problem is given as follows:

$$\begin{aligned}\mathbf{x}^* =&(3.162460616, 3.128331428, 3.094792129, 3.061451495, 3.027929159,\\ &2.993826067, 2.958668718, 2.921842273, 0.494825115, 0.488357110,\\ &0.482316427, 0.476645751, 0.471295508, 0.466230993, 0.461420050,\\ &0.456836648, 0.452458769, 0.448267622, 0.444247010, 0.440382860),\end{aligned}$$

with $f^* = -0.803619104$. Only constraint g_1 is active at this solution. The optimized solution of GAAL method (upto six decimal places of accuracy) is given as follows:

$$\begin{aligned}\mathbf{x}^{\mathrm{GAAL}} =&(3.162461, 3.128331, 3.094792, 3.061451, 3.027929, 2.993826,\\ &2.958669, 2.921842, 0.494825, 0.488357, 0.482316, 0.476645,\\ &0.471296, 0.466231, 0.461420, 0.456837, 0.452459, 0.448268,\\ &0.444247, 0.440383),\end{aligned}$$

and $f^{\mathrm{GAAL}} = -0.803619$. Interestingly, the above GAAL solution is exactly the same as the best-known solution upto at least six decimal places of accuracy. The best, median and worst objective values were -0.803619, -0.803619 and -0.744767, respectively. Here, two out of the 25 runs are not able to find the optimum accurately. But all 25 runs of GAAL were within 10% of the best-known optimum solution. The corresponding Lagrangian multipliers are reported as $\mathbf{u} = (0.074457,\ 0)$, indicating that only constraint g_1 is active at the optimized solution. Interestingly, this agrees with the known result for this problem.

3.4 Problem D: g04

The next problem has five variables and six inequality constraints. The best-known optimum solution (upto nine decimal places of accuracy) is as follows:

$$\mathbf{x}^* = (78, 33, 29.995256026, 45, 36.775812906),$$

with $f^* = -30665.538671783$. Constraints g_1 and g_6 are active at this point. The result of the GAAL method is a follows:

$$\mathbf{x}^{\mathrm{GAAL}} = (78, 33, 29.995256, 45, 36.775813),$$

with $f^{\mathrm{GAAL}} = -30665.538672$ – a solution identical (upto six decimal places of accuracy) to the best-known solution. The best, median and worst objective values obtained from 25 different runs are -30665.538672, -30665.538672 and -30665.538672, respectively, meaning that all 25 runs have found an identical solution. The proposed GAAL method performed quite well in this problem.

3.5 Problem E: g06

This problem has two variables and two inequality constraints. The best-known optimum is given with nine decimal places of accuracy as follows:

$$\mathbf{x}^* = (14.095000000, 0.842960789),$$

and $f^* = -6961.813875580$. Both constraints are active at this optimum. The GAAL optimized solution is presented with six decimal places of accuracy as follows:

$$\mathbf{x}^{\mathrm{GAAL}} = (14.095000, 0.842961),$$

and $f^{\mathrm{GAAL}} = -6961.813876$. Upto six decimal places of accuracy, GAAL solution is identical to that of the best-known solution. The best, median and worst objective values are -6961.813876, -6961.813876 and -6961.813876, respectively, meaning once again that all 25 runs are able to find the same optimized solution. The corresponding Lagrangian multipliers are calculated as $\mathbf{u} = (97739.988401,\ 172760.954414)$, indicating that both constraints are active at the GAAL optimized solution.

3.6 Problem F: g07

This problem has 10 variables and eight inequality constraints. The best-known optimum (with nine decimal places of accuracy) is as follows:

$$\begin{aligned}\mathbf{x}^* =&(2.171996341, 2.363683041, 8.773925739, 5.095984437,\\ &0.990654757, 1.430573929, 1.321644154, 9.828725265,\\ &8.280091589, 8.375926648),\end{aligned}$$

with $f^* = 24.306209068$. Six constraints (g_1 to g_6) are active at this optimum. The result of the GAAL method (with six decimal places of accuracy) is given below:

$$\begin{aligned}\mathbf{x}^{\text{GAAL}} =&(2.171996, 2.363683, 8.773926, 5.095984, 0.990655,\\ &1.430574, 1.321644, 9.828726, 8.280092, 8.375927),\end{aligned}$$

with $f^{\text{GAAL}} = 24.306209$. The best, median and worst objective values for 25 runs are 24.306209, 24.306209 and 24.306209, respectively. It is evident that the proposed method has performed quite well in this problem as well. The corresponding Lagrangian multipliers are computed to be $\mathbf{u}$ = (1.715999, 0.471251, 1.356905, 0.022114, 0.304300, 0.273330, 0, 0), thereby suggesting that the first six constraints are active at the GAAL solution.

3.7 Problem G: g08

This problem has two variables and two inequality constraints. The best-reported optimum solution is given below:

$$\mathbf{x}^* = (1.227971353, 4.245373366),$$

with $f^* = -0.0958250414$. The result of the GAAL method is given as follows:

$$\mathbf{x}^{\text{GAAL}} = (1.227971, 4.245373),$$

with $f^{\text{GAAL}} = -0.095825$. The best, median and worst objective values of 25 runs are -0.095825, -0.095825 and 0.0, respectively. Eight out of the 25 runs are able to locate the optimum accurately. Due to multi-modal nature of this problem, the performance of GAAL in this problem is not as good as that in the previous problems.

3.8 Problem H: g09

This problem has seven variables and four inequality constraints. The best-known optimum (upto nine decimal places of accuracy) is as follows:

$$\begin{aligned}\mathbf{x}^* =&(2.330499351, 1.951372368, -0.477541399, 4.365726249,\\ &-0.624486959, 1.038130994, 1.594226678),\end{aligned}$$

with $f^* = 680.630057374$. Two constraints (g_1 and g_4) are active at this optimum. The GAAL obtained solution (obtained with six decimal places of accuracy) is shown below:

$$\begin{aligned}\mathbf{x}^{\text{GAAL}} =&(2.330499, 1.951372, -0.477541, 4.365726,\\ &-0.624487, 1.038131, 1.594227),\end{aligned}$$

with $f^{\text{GAAL}} = 680.630057$. The best, median and worst function values were 680.630057, 680.630057 and 680.630057, respectively. The proposed method performed quite well in this problem. All 25 runs found the the optimum accurately upto six decimal places of accuracy. The corresponding Lagrangian multipliers are found to be $\mathbf{u}$ = (1.166538, 0, 0, 4.276302), indicating that constraints g_1 and g_4 are active at the obtained optimized solution.

3.9 Problem I: g10

This problem has eight variables and six inequality constraints. The best-known solution with nine decimal places of accuracy is given as follows:

$$\begin{aligned}\mathbf{x}^* =&(579.306685018, 1359.970678079, 5109.970657431, 182.017699630,\\ &295.601173703, 217.982300369, 286.416525928, 395.601173703),\end{aligned}$$

with $f^* = 7049.248020529$. All six constraints are active at this optimum. The GAAL optimized solution is given below:

$$\mathbf{x}^{\mathrm{GAAL}} = (579.306688, 1359.970651, 5109.970682, 182.017700,\\ 295.601173, 217.982300, 286.416527, 395.601173),$$

with $f^{\mathrm{GAAL}} = 7049.248021$. The best, median and worst function values were 7049.248021, 7049.248021 and 7049.248021, respectively, meaning that all 25 runs found the identical solution. The corresponding Lagrangian multipliers are computed to be $\mathbf{u}$ = (1723.265940, 5764.135170, 5218.5992782, 0.008269, 0.010103, 0.010055). This problem has been reported to be a difficult problem to solve due to non-uniform scaling of the constraints [10], as can be seen from the problem formulation given in the appendix. The first three constraints are normalized and the remaining three are not. Interestingly, the GAAL method did not have any difficulty in solving the problem, as in GAAL each constraint is associated with a different multiplier σ_j which is evolved independently. The Lagrange multiplier vector rightly indicates that the first three values take a large u_j value due to having a small constraint value and the last three constraints take small u_j values due to large constraint values. It is the product of Lagrange multiplier value u_j and constraint value g_j that must be of the similar order to the objective function value in order for the composite penalized function $P()$ to provide equal importance to all constraints and the objective function. GAAL is able to adjust all multipliers σ_j accordingly.

3.10 Problem J: g12

This problem has three variables and nine inequality constraints. The best-known solution is given as follows:

$$\mathbf{x}^* = (5, 5, 5),$$

with $f^* = -1$. The GAAL optimized solution is as follows:

$$\mathbf{x}^{\mathrm{GAAL}} = (5.143422, 4.800712, 5.047059),$$

with $f^{\mathrm{GAAL}} = -0.999375$. The best, median and worst function values are -0.999375, -0.999375 and -0.999375, respectively. All 25 runs of GAAL are found to lie within 0.1% of the best-known optimum.

3.11 Problem K: g24

This problem has two variables and two inequality constraints. The best-known optimum is given below:

$$\mathbf{x}^* = (2.329520197, 3.178493074),$$

with $f^* = -5.508013271$. Both constraints are active at this optimum. The GAAL optimized solution is given below:

$$\mathbf{x}^{\mathrm{GAAL}} = (2.329520, 3.178493),$$

with $f^{\mathrm{GAAL}} = -5.508013$. The best, median and worst function values were -5.508013, -5.508013 and -5.508013, respectively. The proposed method performed quite well in this problem. All 25 runs found the the optimum accurately upto six decimal places of accuracy. The corresponding Lagrangian multipliers are computed to be $\mathbf{u}$ = (0.348751, 0.705709), indicating that both constraints are active at this solution.

4 Comparative Results

The GAAL results presented in the previous section were obtained by setting a termination condition that checked objective value of two consecutive local-searched solutions in Step 2(d). If the absolute difference in objective values were less than or equal to 10^{-4}, the run was terminated. Importantly, in these runs, the termination was not made based on a comparison with best-known solutions. Despite, the GAAL method was able to converge very close to the correct optimum solution in all problems and in almost all 25 runs. This indicates the robustness and efficiency of the proposed method.

In this section, we are interested in comparing the performance of our GAAL method with some well-known evolutionary constraint handling methods. Three state-of-the-art methods are chosen for this purpose. They are taken from references [25], [21] and [3]. The termination condition used in these studies were slightly different. If a solution having an objective value within 10^{-4} from the best-known objective value is found, a run is terminated. This requires one to know the best-known solutions and thus the approach is applicable for solving test problems only. Nevertheless, we compare our results (in which the knowledge of the best-known solution is not utilized) with the existing results reported in [25], [21] and [3] in Table 1. It can be observed that the GAAL achieves the best-known optimum with the set accuracy level in smaller number of function evaluations than all three competing evolutionary methods in almost all problems. For problem g08, Takahama and Sakai's algorithm [21] performs the best and GAAL performs the worst. For problem g02, only two out of 25 runs comes close to 0.1% of the best-known objective value. Both problems g02 and g08 have a periodic and multimodal objective function and as discussed elsewhere [10], the multi-modality causes difficulty to penalty based approaches. The three competing evolutionary methods used a differential evolution (DE) which uses a difference between two variable vectors to create a child solution. Particularly, for periodic objective functions in which multiple optima lie in a periodic manner on the search space, DE based methods are expected to perform better. However, the superior performance of GAAL in most other problems having a large number of variables, a large number of constraints, non-uniform scaling of constraints and other complexities, remains as an interesting outcome of this study.

Table 1: Comparison of function evaluations needed by GAAL and by other best-known evolutionary algorithms.

Prob	Zavala et al.[25]			Takahama & Sakai[21]			Brest[3]			Proposed Method		
	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
g01	80776	90343	96669	18594	19502	**19971**	51685	55211	57151	**12617**	**15870**	34717
g02	87419	93359	99654	108303	114347	129255	175090	226789	253197	**15849**	**38937**	**66785**
g04	3147	103308	110915	12771	13719	14466	56730	62506	67383	**1861**	**3611**	**4111**
g06	95944	109765	130293	5037	5733	**6243**	31410	34586	37033	**1005**	**2755**	12355
g07	114709	138767	208751	60873	67946	75569	184927	197901	221866	**10408**	**20219**	**30218**
g08	2270	4282	5433	**621**	**881**	**1173**	1905	4044	4777	4605	100606	100809
g09	94593	103857	119718	19234	21080	21987	79296	89372	98062	**2953**	**6450**	**8205**
g10	109243	135735	193426	87848	92807	107794	203851	220676	264575	**9371**	**10578**	**16175**
g12	482	6158	9928	2901	4269	5620	**364**	6899	10424	1209	**1705**	**1957**
g18	97157	107690	124217	46856	57910	60108	139131	169638	191345	**11461**	**14161**	**26306**
g24	11081	18278	633378	1959	2451	**2739**	9359	12844	14827	**1005**	**1505**	3911

Finally, we compare our proposed GAAL method with the classical AL method solved using MATLAB's `fmincon` routine. In this approach, each σ_j is initialized to zero. Thereafter, the penalized function (Equation 2) is formulated and solved using the `fmincon` routine. A random solution is used as an initial solution. Thus, to make the comparison fair, the algorithm is run 25 times and each time starting with a different initial random solution. Based on the optimum solution obtained for the first penalized function, the multipliers are updated using Equation 3 and another optimization run is performed to solve the revised penalized function. This time, the optimal solution obtained from the previous optimization of penalized function is used as the initial solution. An identical termination condition (a solution having at most 10^{-4} difference between optimal objective values of two consecutive penalized functions) is used here. However, if such a scenario is not achieved within a maximum of 5,000 iterations, the final solution at the end of 5,000 iteration is reported. Note that in such an event, the reported solution can be infeasible as well. Table 2 shows the comparison of the results of the proposed GAAL method with the results obtained from the above classical AL method. It can be observed that in only a few cases the classical AL approach is able to find a solution close to the best-known solution and surprisingly in many cases ends up getting stuck to an infeasible solution. Only problem g09 is solved in 25 out of 25 runs. This table clearly indicates the usefulness of a more global and population-based genetic algorithm as an efficient constrained optimization algorithm that can be used through the augmented Lagrangian approach. The use of a seamless and self-adaptive penalty parameter and multiplier update and an embedding of an occasional local search operation to provide convergence property are all necessary ingredients that have helped the overall algorithm to work well on a variety of difficult constrained optimization problems.

4.1 An Engineering Problem: Welded Beam Design

Finally, GAAL is applied to the well-known welded beam design problem presented in [7]. This problem has four variables and five inequality constraints. For a welded beam, the cost of fabrication is required to be minimized subject to two stress constraints, one physical constraint, one buckling constraint, and one deflection constraint. The physical constraint states that two of the variables (h and t) must be related as $h \leq b$.

We set the population size as 50 and maximum generations as 500 for this problem. The best-known optimum (with four decimal places of accuracy) for this problem is as follows:

$$(h, \ell, t, b)^* = (0.2444, 6.2187, 8.2915, 0.2444),$$

having a cost objective value of $f^* = 2.38117$ units. The first four constraints are active at this optimum.

The best GAAL solution is found to be as follows:

$$(h, \ell, t, b)^{\text{GAAL}} = (0.244369, 6.218607, 8.291472, 0.244369),$$

with $f^{\text{GAAL}} = 2.381134$ units – an 0.001% improvement from the best-known solution. Despite the introduction of this problem to classical optimization literature in eighties [19], to evolutionary computation literature in nineties [5], and extensive studies made thereafter, the GAAL obtained solution is better than the best-reported solution to this problem so far.

The best, median and worst function values over 25 runs are 2.381134, 2.381134 and 2.381134 units, respectively, thereby depicting the robustness of the proposed algorithm. The minimum, median and maximum number of function evaluations required to achieve termination in all 25 runs are 3,909, 5,911 and 10,515, respectively. These numbers are small compared to that needed in other evolutionary constrained optimization studies – 320,000 needed in [7] and a median of 8,574 in a recent study [10]. The corresponding Lagrangian multipliers are reported as $\mathbf{u} = (1.235694,\ 1.069852,\ 13.081068,\ 0.017638,\ 0)$, meaning that the first four constraints are active at this solution. Interestingly, the above GAAL solution requires that $h = b$, thereby making the corresponding (third) constraint active at the GAAL solution. A large value of u_3 indicates that the third constraint ($h \leq b$) is most sensitive to the optimum solution, that is, a small change

Table 2: Comparison of function evaluations needed by GAAL and a classical AL method. Successful run means result is within 0.1% of the best-known optimum and is feasible. Infeasible results are shown in italics.

Prob.	Best-known optimum	Proposed method, GAAL						
		f^*			Function evaluations			Succ.
		Best	Median	Worst	Min	Median	Max	runs
g01	-15	-15	-15	-12.453125	12617	15870	34717	24
g02	-0.803619	-0.803619	-0.803619	-0.744767	15849	38937	66785	2
g04	-30665.538671	-30665.538672	-30665.538672	-30665.538672	1861	3611	4111	25
g06	-6961.813876	-6961.813876	-6961.813876	-6961.813876	1005	2755	12355	25
g07	24.306209	24.306209	24.306209	24.306209	10408	20219	30218	25
g08	-0.095825	-0.095825	-0.095825	0	4605	100606	100809	8
g09	680.630057	680.630057	680.630057	680.630057	2953	6450	8205	25
g10	7049.248021	7049.248021	7049.248021	7049.248021	9371	10578	16175	25
g12	-1	-0.999375	-0.999375	-0.999375	1209	1705	1957	25
g18	-0.866025	-0.866025	-0.866025	-0.674981	11461	14161	26306	24
g24	-5.508013	-5.508013	-5.508013	-5.508013	1005	1505	3911	25
Prob.	Best-known optimum	Classical ALM						
		f^*			Function evaluations			Succ.
		Best	Median	Worst	Min	Median	Max	runs
g01	-15	-7.697937	-6	-6	225	529	1313	0
g02	-0.803619	-0.411793	-0.213328	-0.186381	455	1084	2247	0
g04	-30665.538671	*-32217.431037*	-30665.541418	-29083.524056	391	546	80381	6
g06	-6961.813876	-6961.813662	-6961.760172	-4518.090906	925	962	152087	17
g07	24.306209	*24.304042*	*24.30512*	24.313922	8106	160203	160203	1
g08	-0.095825	*-58.385784*	-0.025741	0.024911	89	89	102790	0
g09	680.630057	680.629056	680.659357	680.982856	2522	2920	53700	25
g10	7049.248021	*2100*	7049.312618	7049.493868	8125	19113	2779221	14
g12	-1	*-1*	*-1*	-0.25	198	30117	30738	1
g18	-0.866025	-0.695361	0	0	1815	2570	9740	0
g24	-5.508013	*-5.508022*	-5.507574	-3	16	145	100750	5

in this constraint will make a large sacrifice in the optimal objective value. Thus, adhering to dimensions $h = b = 0.244369$ inches is an important design goal for this problem. The fifth constraint involves a deflection constraint which does not get satisfied with the equality sign, as an arbitrary large threshold (0.25 inches) on allowable deflection is chosen in the formulation. It is clear from the results that not only GAAL is able to find an improved solution, the method is also reliable and computationally fast.

5 Conclusions

In this paper, we have proposed a hybrid and self-adaptive genetic algorithm that follows the principle of augmented Lagrangian (AL) method in solving constrained optimization problems. The proposed algorithm (GAAL) updates both the associated penalty parameter (which is usually kept fixed in a classical AL method) and multipliers for all constraints in a seamless manner. Occasionally, the population-best solution is modified by applying a local search using a well-known numerical optimization algorithm to provide convergence property to the overall algorithm. The proposed GAAL procedure has shown to work better and more reliably than three competing evolutionary approaches in most of the standard test problems of this study in terms of computational efforts. The obtained solutions are also found to be remarkably accurate (upto six decimal places) to the best-known results. The method has also shown to work well on a practical design problem. A comparison with the classical AL method has also revealed the importance of using a genetic algorithm as an optimization method for constraint handling.

This method takes advantage of GA's population approach and classical optimization algorithm's convergence properties, and exploits the algorithmic advantage of the augmented Lagrangian approach. The extent of an optimization run for a fixed value of multipliers is self-adjusted and importantly the penalty parameter value, which is usually kept fixed in a classical AL approach, is also adjusted in a self-adaptive manner. A combination of all above has made the overall GAAL method to be accurate, computationally fast, and reliable over multiple runs on most problems of this study.

A by-product of the proposed methodology is that the Lagrange multipliers corresponding to constraint functions have also been obtained without any additional effort, thereby giving us information about the influence and sensitivity of individual constraints on the optimized solution. Despite some recent interests on finding Lagrange multipliers [22] for GA based approaches, evolutionary constrained optimization researchers have not paid much attention towards this important and pragmatic aspect of computing Lagrange multipliers. Evolutionary algorithms have potential for this purpose and more EAs now must be devised for not only finding the optimal solution, but also to compute Lagrange multipliers efficiently. This paper remains as an important step in this direction.

Acknowledgments

This study is supported by Academy of Finland under grant 133387.

References

[1] Amstutz, S.: Augmented lagrangian for cone constrained topology optimization. Computational Optimization and Applications **49**(1), 101–122 (2011)

[2] Birgin, E.G., Martnez, J.M.: Augmented lagrangian method with nonmonotone penalty parameters for constrained optimization. Computational Optimization and Applications (2011). DOI: 10.1007/s10589-011-9396-0

[3] Brest, J.: Constrained Real-Parameter Optimization with ε Self-Adaptive Differential Evolution, pp. 73–94. springer (2009)

[4] Coello, C.A.C.: Treating objectives as constraints for single objective optimization. Engineering Optimization **32**(3), 275–308 (2000)

[5] Deb, K.: Optimal design of a welded beam structure via genetic algorithms. AIAA Journal **29**(11), 2013–2015 (1991)

[6] Deb, K.: Optimization for Engineering Design: Algorithms and Examples. New Delhi: Prentice-Hall (1995)

[7] Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering **186**(2–4), 311–338 (2000)

[8] Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems **9**(2), 115–148 (1995)

[9] Deb, K., Agrawal, S.: A niched-penalty approach for constraint handling in genetic algorithms. In: Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA-99), pp. 235–243. Springer-Verlag (1999)

[10] Deb, K., Datta, R.: A fast and accurate solution of constrained optimization problems using a hybrid bi-objective and penalty function approach. In: Proceedings of the IEEE World Congress on Computational Intelligence (WCCI-2010), pp. 165–172 (2010)

[11] Homaifar, A., Lai, S.H.V., Qi, X.: Constrained optimization via genetic algorithms. Simulation **62**(4), 242–254 (1994)

[12] Joines, J.A., Houck, C.R.: On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GAs. In: Proceedings of the International Conference on Evolutionary Computation, pp. 579–584 (1994)

[13] Kowalewski, A., Lasiecka, I., Sokolowski, J.: Sensitivity analysis of hyperbolic optimal control problems. Computational Optimization and Applications (2011). DOI: 10.1007/s10589-010-9375-x

[14] Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello, C.A.C., Deb, K.: Special session on constrained real-parameter optimization (http://www.ntu.edu.sg/home/epnsugan/) (2006)

[15] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Berlin: Springer-Verlag (1992)

[16] Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation Journal **4**(1), 1–32 (1996)

[17] Pillo, G.D., Liuzzi, G., Lucidi, S., Palagi, L.: A truncated newton method in an augmented lagrangian framework for nonlinear programming. Computational Optimization and Applications **45**(2), 311–352 (2010)

[18] Rao, S.S.: Genetic algorithmic approach for multiobjective optimization of structures. In: Proceedings of the ASME Annual Winter Meeting on Structures and Controls Optimization, vol. 38, pp. 29–38 (1993)

[19] Reklaitis, G.V., Ravindran, A., Ragsdell, K.M.: Engineering Optimization Methods and Applications. New York : Wiley (1983)

[20] Srivastava, S., Deb, K.: A genetic algorithm based augmented lagrangian method for computationally fast constraint optimization. In: Proceedings of the First International Conference on Swarm, Evolutionary and Memetic Computing (SEMCCO 2010), pp. 182–189. Berlin: Springer-Verlag (2010)

[21] Takahama, T., Sakai, S.: Constrained optimization by the ϵ constrained differential evolution with gradient-based mutation and feasible elites. In: 2006 IEEE Congress on Evolutionary Computation, pp. 1–8. Piscatway: IEEE Press (2006)

[22] Tulshyan, R., Arora, R., Deb, K., Dutta, J.: Investigating ea solutions for approximate kkt conditions for smooth problems. In: Proceedings of Genetic and Evolutionary Algorithms Conference (GECCO-2010), pp. 689–696 (2010)

[23] Volkwein, S.: Application of the augmented lagrangian-sqp method to optimal control problems for the stationary burgers equation. Computational Optimization and Applications **16**(1), 57–81 (2010)

[24] Y. Zhou Y. Li, J.H., Kang, L.: Multi-objective and mgg evolutionary algorithm for constrained optimization. In: Proceedings of Congress on Evolutionary Computation, pp. 1–5 (2003)

[25] Zavala, A.E.M., Aguirre, A.H., Diharce, E.R.V.: Continuous Constrained Optimization with Dynamic Tolerance Using the COPSO Algorithm, pp. 1–24. Heidelberg, Germany: Springer (2009)

[26] Zhou, Y.Y., Yang, X.Q.: Augmented lagrangian functions for constrained optimization problems. Journal of Global Optimization (2010). DOI: 10.1007/s10898-011-9688-z

A Problem Formulations

These problems are taken from [14].

A.1 Problem g01

The problem is given as follows:

$$\begin{array}{rl} \text{min.} & f(\mathbf{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 + 5\sum_{i=5}^{13} x_i, \\ \text{s.t.} & g_1(\mathbf{x}) \equiv 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0, \\ & g_2(\mathbf{x}) \equiv 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0, \\ & g_3(\mathbf{x}) \equiv 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0, \\ & g_4(\mathbf{x}) \equiv -8x_1 + x_{10} \leq 0, \\ & g_5(\mathbf{x}) \equiv -8x_2 + x_{11} \leq 0, \\ & g_6(\mathbf{x}) \equiv -8x_3 + x_{12} \leq 0, \\ & g_7(\mathbf{x}) \equiv -2x_4 - x_5 + x_{10} \leq 0, \\ & g_8(\mathbf{x}) \equiv -2x_6 - x_7 + x_{11} \leq 0, \\ & g_9(\mathbf{x}) \equiv -2x_8 - x_9 + x_{12} \leq 0, \end{array} \tag{10}$$

where $0 \leq x_i \leq 1$ for $i = 1, 2, \ldots, 9$, $0 \leq x_i \leq 100$ for $i = 10, 11, 12$, and $0 \leq x_{13} \leq 1$.

A.2 Problem g02

The problem is given as follows:

$$\begin{array}{rl} \text{min.} & f(\mathbf{x}) = -\left| \frac{\sum_{i=1}^{20} \cos^4(x_i) - 2\Pi_{i=1}^{20} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{20} i x_i^2}} \right|, \\ \text{s.t.} & g_1(\mathbf{x}) \equiv 0.75 - \Pi_{i=1}^{20} x_i \leq 0, \\ & g_2(\mathbf{x}) \equiv \sum_{i=1}^{20} x_i - 150 \leq 0, \\ & 0 \leq x_i \leq 10, \quad i = 1, 2, \ldots, 20. \end{array} \tag{11}$$

A.3 Problem g04

The problem is given as follows:

$$\begin{array}{rl}\text{min.} & f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141, \\ \text{s.t.} & g_1(\mathbf{x}) \equiv 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 \\ & \quad -0.0022053x_3x_5 - 92 \le 0, \\ & g_2(\mathbf{x}) \equiv -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 \\ & \quad +0.0022053x_3x_5 \le 0, \\ & g_3(\mathbf{x}) \equiv 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 \\ & \quad +0.0021813x_3^2 - 110 \le 0, \\ & g_4(\mathbf{x}) \equiv -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 \\ & \quad -0.0021813x_3^2 + 90 \le 0, \\ & g_5(\mathbf{x}) \equiv 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 \\ & \quad +0.0019085x_3x_4 - 25 \le 0, \\ & g_6(\mathbf{x}) \equiv -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 \\ & \quad -0.0019085x_3x_4 + 20 \le 0, \\ & 78 \le x_1 \le 102, 33 \le x_2 \le 45, 27 \le (x_3, x_4, x_5) \le 45. \end{array} \tag{12}$$

A.4 Problem g06

The problem is given as follows:

$$\begin{array}{rl}\text{min.} & f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3, \\ \text{s. t.} & g_1(\mathbf{x}) \equiv -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0, \\ & g_2(\mathbf{x}) \equiv (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0, \\ & 13 \le x_1 \le 100, \quad 0 \le x_2 \le 100. \end{array} \tag{13}$$

A.5 Problem g07

The problem is given as follows:

$$\begin{array}{l}\text{min. } f(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 \\ \quad +4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 \\ \quad +2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45, \\ \text{s.t.} \\ g_1(\mathbf{x}) \equiv -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0, \\ g_2(\mathbf{x}) \equiv 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0, \\ g_3(\mathbf{x}) \equiv -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0, \\ g_4(\mathbf{x}) \equiv 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_2^3 - 7x_4 - 120 \le 0, \\ g_5(\mathbf{x}) \equiv 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0, \\ g_6(\mathbf{x}) \equiv x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0, \\ g_7(\mathbf{x}) \equiv 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0, \\ g_8(\mathbf{x}) \equiv -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0, \\ -10 \le x_i \le 10, \quad i = 1, 2, \ldots, 10. \end{array} \tag{14}$$

A.6 Problem g08

The problem is given as follows:

$$\begin{array}{rl}\text{min.} & f(\mathbf{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}, \\ \text{s.t.} & g_1(\mathbf{x}) \equiv x_1^2 - x_2 + 1 \le 0, \\ & g_2(\mathbf{x}) \equiv 1 - x_1 + (x_2 - 4)^2 \le 0, \\ & 0 \le x_i \le 10, \quad i = 1, 2. \end{array} \tag{15}$$

A.7 Problem g09

The problem is given as follows:

$$\begin{aligned}
\text{min.} \quad & f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\
& \quad +10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7, \\
\text{s.t.} \quad & g_1(\mathbf{x}) \equiv -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0, \\
& g_2(\mathbf{x}) \equiv -282 + 7x_1 + 3x_2 + 10x_2^3 + x_4 - x_5 \leq 0, \\
& g_3(\mathbf{x}) \equiv -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0, \\
& g_4(\mathbf{x}) \equiv 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0, \\
& -10 \leq x_i \leq 10, \quad i = 1, 2, \ldots, 7.
\end{aligned} \tag{16}$$

A.8 Problem g10

The problem is given as follows:

$$\begin{aligned}
\text{min.} \quad & f(\mathbf{x}) = x_1 + x_2 + x_3, \\
\text{s.t.} \quad & g_1(\mathbf{x}) \equiv -1 + 0.0025(x_4 + x_6) \leq 0, \\
& g_2(\mathbf{x}) \equiv -1 + 0.0025(x_5 + x_7 - x_4) \leq 0, \\
& g_3(\mathbf{x}) \equiv -1 + 0.01(x_8 - x_5) \leq 0, \\
& g_4(\mathbf{x}) \equiv -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0, \\
& g_5(\mathbf{x}) \equiv -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0, \\
& g_6(\mathbf{x}) \equiv -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0, \\
& 100 \leq x_1 \leq 10000, 1000 \leq (x_2, x_3) \leq 10000, \\
& 10 \leq (x_4, \ldots, x_8) \leq 1000.
\end{aligned} \tag{17}$$

A.9 Problem g12

The problem is given as follows:

$$\begin{aligned}
\text{min.} \quad & f(\mathbf{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100, \\
\text{s.t.} \quad & g_i(\mathbf{x}) \equiv (x_1 - p_i)^2 + (x_2 - q_i)^2 + (x_3 - r_i)^2 - 0.0625 \leq 0, \\
& \qquad \forall i = 1, 2, \ldots, 9, \\
& 0 \leq x_i \leq 100, \quad i = 1, 2, 3.
\end{aligned} \tag{18}$$

A.10 Problem g18

The problem is given as follows:

$$\begin{aligned}
\text{min.} \quad & f(\mathbf{x}) = -0.5\left(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7\right), \\
\text{s.t.} \quad & g_1(\mathbf{x}) \equiv x_3^2 + x_4^2 - 1 \leq 0, \\
& g_2(\mathbf{x}) \equiv x_9^2 - 1 \leq 0, \\
& g_3(\mathbf{x}) \equiv x_5^2 + x_6^2 - 1 \leq 0, \\
& g_4(\mathbf{x}) \equiv x_1^2 + (x_2 - x_9)^2 - 1 \leq 0, \\
& g_5(\mathbf{x}) \equiv (x_1 - x_5)^2 + (x_2 - x_6) \leq 0, \\
& g_6(\mathbf{x}) \equiv (x_1 - x_7)^2 + (x_2 - x_8) \leq 0, \\
& g_7(\mathbf{x}) \equiv (x_3 - x_5)2 + (x_4 - x_6)^2 - 1‘0, \\
& g_8(\mathbf{x}) \equiv (x_3 - x_7)^2 + (x_4 - x_8)^2 - \leq 0, \\
& g_9(\mathbf{x}) \equiv x_2 + (x_8 - x_9)^2 - 1 \leq 0, \\
& g_{10}(\mathbf{x}) \equiv x_2x_3 - x_1x_4 \leq 0, \\
& g_{11}(\mathbf{x}) \equiv -x_3x_9 \leq 0, \\
& g_{12}(\mathbf{x}) \equiv x_5x_9 \leq 0, \\
& g_{13}(\mathbf{x}) \equiv x_6x_7 - x_5x_8 \leq 0, \\
& -10 \leq x_i \leq 10, \quad i = 1, 2, \ldots, 8, \\
& 0 \leq x_9 \leq 20.
\end{aligned} \tag{19}$$

A.11 Problem g24

The problem is given as follows:

$$\begin{array}{ll} \text{min.} & f(\mathbf{x}) = -x_1 - x_2, \\ \text{s.t.} & g_1(\mathbf{x}) \equiv -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0, \\ & g_2(\mathbf{x}) \equiv -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x2 - 36 \leq 0, \\ & 0 \leq x_1 \leq 3, \quad 0 \leq x_2 \leq 4. \end{array} \tag{20}$$

A.12 Problem Weld

The problem is given as follows ($\mathbf{x} = (h, l, t, b)^T$) [19, 5]:

$$\begin{array}{ll} \text{min.} & f_1(\mathbf{x}) = 1.10471h^2l + 0.04811tb(14.0 + l), \\ \text{s. t.} & g_1(\mathbf{x}) \equiv 13,600 - \tau(\mathbf{x}) \geq 0, \\ & g_2(\mathbf{x}) \equiv 30,000 - \sigma(\mathbf{x}) \geq 0, \\ & g_3(\mathbf{x}) \equiv b - h \geq 0, \\ & g_4(\mathbf{x}) \equiv P_c(\mathbf{x}) - 6,000 \geq 0, \\ & g_5(\mathbf{x}) \equiv 0.25 - \delta(\mathbf{x}) \geq 0, \\ & 0.125 \leq h, b \leq 5, \\ & 0.1 \leq l, t \leq 10, \end{array} \tag{21}$$

where,

$$\begin{array}{rcl} \tau(\mathbf{x}) & = & \sqrt{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')/\sqrt{0.25(l^2 + (h+t)^2)}}, \\ \tau' & = & \dfrac{6,000}{\sqrt{2}hl}, \\ \tau'' & = & \dfrac{6,000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2[0.707hl(l^2/12 + 0.25(h+t)^2)]}, \\ \sigma(\mathbf{x}) & = & \dfrac{504,000}{t^2b}, \quad \delta(\mathbf{x}) = \dfrac{2.1952}{t^3b}, \\ P_c(\mathbf{x}) & = & 64,746.022(1 - 0.0282346t)tb^3. \end{array}$$