Optimization of Constraint problems using Bracket Operator penalty method

Rohan Chinchkar
M.tech, Machine Design
IIT Guwahati
Guwahati,India
c.rohan@iitg.ac.in

Shivam Sinha
M.Tech, Machine Design
IIT Guwahati
Guwahati,India
shivamsinha@iitg.ac.in

***Abstract*— Constraint Optimization Problems having single global optima are solved with bracket operator penalty method using generic C program. The conjugate direction method helps to convert obtained unconstrained function into single variable optimization function and further it is handled by combined bounding phase and secant method algorithm to perform the unidirectional search.**

Keywords—optimization, constraint, conjugate, multivariable, bounding, secant, algorithm, unidirection.

I. INTRODUCTION

In optimization, constraint function method is used to convert given constraint function into unconstraint function so that we can use multivariable optimization techniques to find out the optimum point of that function. Basically one popular method known as bracket penalty method is extensively used in constraint optimization method. It adds penalty each time when algorithm violates any of the constraint i.e. point goes into the infeasible region, this helps to find the solution in feasible region only.

Along with the penalty function method, we need to use algorithm which can solve the unconstrained function and for that we have used conjugate direction method which is one of the most successful method to find optimal point of a multivariable function. Also we have used bounding phase and secant method to accurately perform the unidirectional search as required for the conjugate direction method. All of this is performed by a 'C' program which has been made for excellent accuracy and effectiveness of the optimization method

The given problems are written in the separate objective function using switch case and its constraints are also written in separate function to maintain the simplicity for the new problems. These two functions are connected by another function called as penalty function for the easy access for the user. User have to only enter the problem number and algorithms will be executed automatically by the program. After solving the problem, the program automatically generates a 'txt file' and stores all the obtained results with iterative data into it.

A constrained optimization problem comprises an objective function together with a number of equality and inequality constraints. Often lower and upper bounds on design or decision variables are also specified. Considering that there are N decision or design variables, we write a constrained problem as follows:

Minimize f(x)

subject to, $g_j(x) \geq 0$, j = 1, 2, . . . , J;

$$h_k(x) = 0, \quad k = 1, 2, \ldots, K; \qquad (1)$$

$$xi^{(L)} \leq xi \leq xi^{(U)}, \quad i = 1, 2, \ldots, N.$$

A solution satisfying all constraints and bounds on variables is called feasible solution. Otherwise, it is an infeasible solution.

Nonlinear programming problem (NLP): None of the functions in eq. (1) are assumed to be linear.

Bracket Operator Penalty Method:

$$\Omega = R\langle g(x)\rangle^2$$

where $\langle\alpha\rangle=\alpha$, when α is negative; zero otherwise. Since it assigns positive value to infeasible point, it is an exterior penalty term. It starts with small value of R which increases gradually.

II. ALGORITHM

A. *Penalty Function Method:*

Step 1: Choose two termination parameters $\epsilon1$, $\epsilon2$, an initial solution x(0), a penalty term Ω, and an initial penalty parameter R(0). Choose a parameter c to update R such that 0 < c < 1 is used for interior penalty terms and c > 1 is used for exterior penalty terms. Set t = 0.

Step 2: Form P(x(t),R(t)) = f(x(t)) + Ω(R(t), g(x(t)), h(x(t))).

Step 3: Starting with a solution x(t), find x(t+1) such that P(x(t+1),R(t)) is minimum for a fixed value of R(t).

use $\epsilon1$ to terminate the unconstrained search.

Step 4: Is $|P(x(t+1),R(t)) - P(x(t),R(t-1))| \leq \epsilon2$? If yes, set xT = x(t+1) and Terminate; Else go to Step 5.

Step 5: Choose R(t+1) = cR(t). Set t = t + 1 and go to Step 2.

B. *Powell's Conjugate Direction Method:*

Step 1: Select proper limits, initial guess and required accuracy

Step 2: Set each direction to unit independent direction

Step 3: Calculate next point with X1 = X0 + α*S1

Step 4: Put this in the function and calculate value of α for min(F(α))

Step 5: After X1, Calculate next point w.r.t. previous point along the next direction

Step 6: Find all minimum points along the respective search directions

Step 7: Perform search along the 1st direction again

Step 8: Find vector d = Xn – X1 and check || d || < ε1

Step 9: If yes then shift the direction in forward order and assign S1=d

Step 10: Follow steps 3 to 9 until we get || d || > ε1,

|| Si .Si+1 || = 0

C. Bounding Phase Method :

Step 1: Select the values of upper and lower limits (a, b)

Step 2: Assume initial guess (x1) and required increment (Delta) .

Step 3: Find values of x2 = x1-Delta, x3 = x1 + Delta, f(x1), f(x2), f(x3) .

Step 4: Compare the values of f(x1), f(x2), f(x3) and select the sign of (Delta) according to f(x2) > f(x1) > f(x3) as positive, else negative

Step 5: Then find new x2 by x2 = x1 + 2k (Delta), where k is the number of iterations

D. Secant Method:

Step 1: Get limits from bounding phase method (a, b)

Step 2: Now calculate f'(a), f'(b) and compare these values Step 3: if these are not equal then proceed further to calculate

$z = x2 - f'2 * (x2 - x1)(f'2 - f'1)$

Step 4: Now, if f'(z) < 0 then (x1 = z) , else (x2 = z)

Step 5: follow these steps until we get abs(f'(z)) < e,

where e is error.

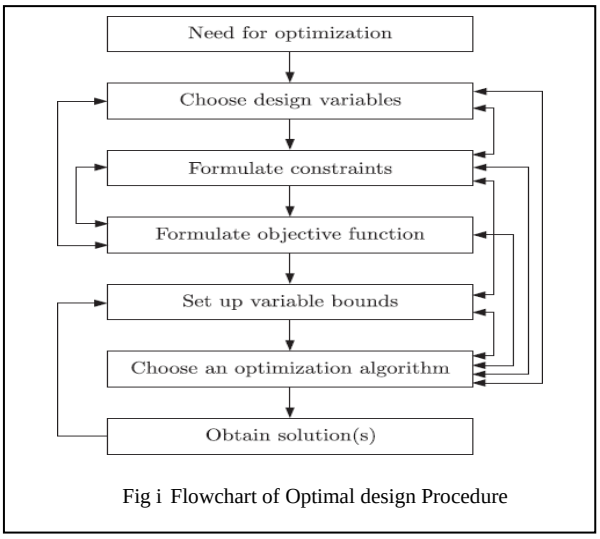

Fig i Flowchart of Optimal design Procedure

III. COMPUTATIONAL COMPLEXITY OF ALGORITHM

What we found that, converting a constrained function into unconstrained one and solving it, is not a difficult task because of available algorithms of various methods. But in this project, we faced some trouble while building the appropriate code for given problem. The code is generated to give the maximum efficiency to find the solution of a constrained function problem. At this stage finding the solution is easy for a smooth function having only single optimum point but if it has multiple local optimum points then it is very difficult to tackle them and get the global optima at this level of algorithms.

Since the program has to go through lot of iterations and sequences to get the solution with desired accuracy, it performs huge number of function evaluation. Also we need to check at every iteration that different search directions used for conjugate direction method are linearly independent or not, if it is not then we have to change the directions little bit. For checking dependency of the search directions, we have used gauss elimination method of matrices.

As per the difficulty level of algorithm, it is capable of handling any constraint optimization problem with smooth curve and having only single optimum point very effectively.

IV. RESULTS AND DISCUSSIONS

We have generated the C program for solving constraint optimization problems, for that we have used 'Dev-C++' application of windows which allows us to create our code, compile it and run that code. In the program we have used arrays of different dimensions to store and execute the operation. 'X' and 'S' are used for variables and search directions of dimension 'c' which is number of variables. Similarly we used the other parameters exactly as mentioned in the algorithms to maintain standards of our code.

For given three problems, we observed that function involves some local optimum points within the prescribed range and sometimes program catches these points. Still we have fair chances of getting correct results if we choose initial guess correctly. We observed that code find outs latest crest/trough within the range from the initial point and accordingly converges to the nearest optima. From the obtained results, we have plotted convergence graph (Function value vs. number of iterations), function evaluations vs. number of iterations and also compared the results with the known correct results and formulated the table named TABLE I.

Results of first two problems are quite satisfactory since they have given correct solutions for almost all the time even having some local optimum points. But in problem number 3, we faced some issue because of multiple local optimum points and large number of constraints (8 constraints), because of it, program was unable to tackle those points and find the global optimum solution. This can be avoided by modification in the code using advanced algorithms of optimization to improve the effectiveness while handling local optimum points.

Table I. Comparison of function values

Fun. Values	**Problem Number**		
	1	***2***	***3***
Best	-6961.80187557	0.09582504	7401.39435414
Worst	-6961.80130498	0.02726283	3654.62524051
Mean	-6961.80170195	0.075444474	4608.91732947
Median	-6961.80184268	0.09582504	4530.20160129
Mode	-6961.80180	0.09582504	4322.68068873
Exact Fun Value	-6961.80198073	0.09582504	7049.3307
Std Deviation	0.0003179615	0.03721282	2704.449673

From above table we can say that the best value given by the program is very close to the known correct value of global optimum point.

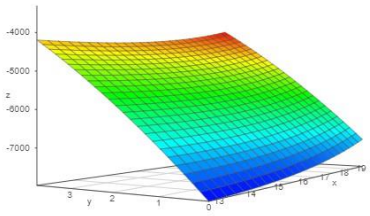

Fig ii Surface plot of function for problem 1

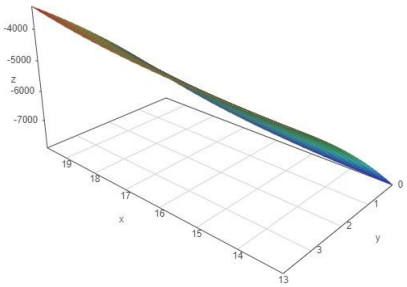

Fig iii Surface plot of function for problem 1

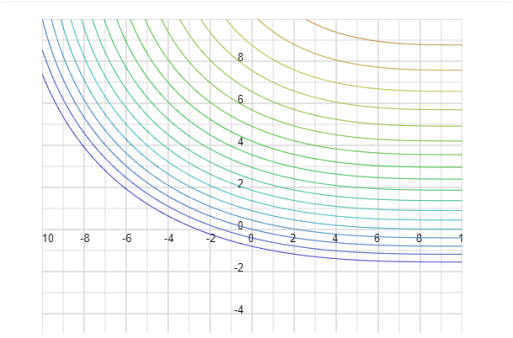

Fig iv Contour for problem 1

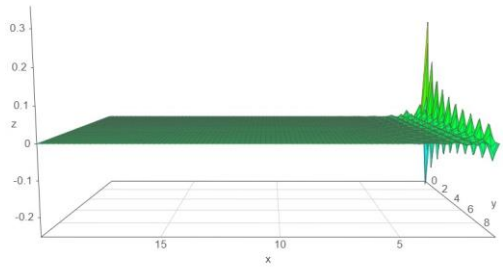

Fig v Surface plot of function for problem 2

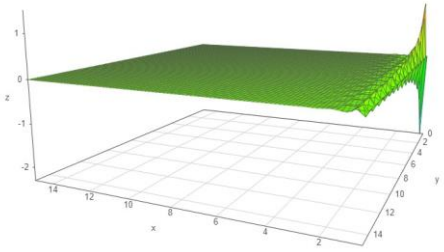

Fig vi surface plot of function for problem 2

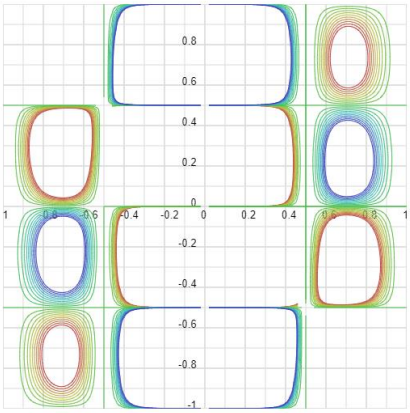

Fig vii Contour plot for problem 2

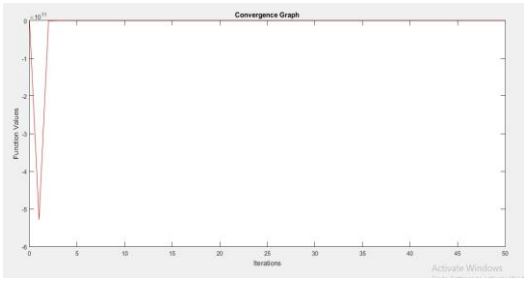

Fig vi Convergence graph for problem 1

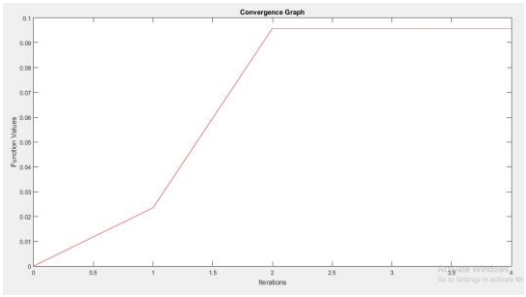

Fig vii Convergence graph for problem 2

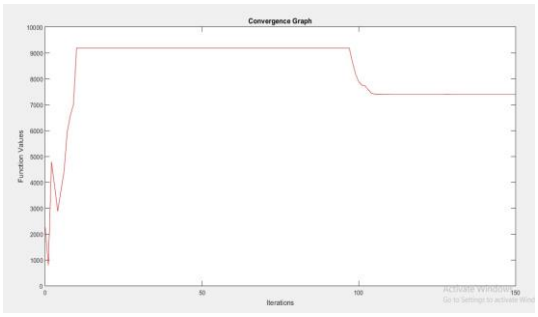

Fig viii Convergence graph for problem 3

We have attached contour plots and surface plots for problem1 and 2 and convergence plots for all three problems. By observing convergence plots, we understood that problem first will converge to the global optima irrespective of the constraints. That is because of very small value of penalty function (R). Further it will consider the constraints violations as the value of R keeps on increasing and will converge to the optimal solution in the feasible solution. Hence convergence graph is not like the unconstrained problems and is somewhat looking disturbed.

Conjugate direction performed very well for independent set of directions, but at some point directions became dependent and we have to reinitialize the search direction. This part is also done automatically by the program and it is made to do so.

Another thing was observed during the experimentation that bounding phase quickly catches the optimal region but in case of secant method, it took lot of loops to locate the optimal point for the unidirectional search in the respective region. This might be because of the slope near the optimal point gets flatten and to locate optima by secant method, it finds trouble due to dependency on slope while performing the operation.

While solving problem 3, the program was giving different solution values for every new initial guess and this might be because of multiple constraints and multiple local optima. At this stage of learning we are unable to locate the minima for such kind of problem (3). We have to use advanced optimization algorithms to handle local optimal points and get the exact optimal point.

The program is made such that any constraint problem can be solved using this code, the only condition is that the function should be without local optimum points i.e. smooth type of surface

V. Conclusions

By performing various experiments with different initial guesses, we can confidently say that the generated code is capable of finding optimal solution of any smooth constraint function. Also bracket penalty method is much effective method while solving constraint optimization problems as it makes the function to consider violation of constraints.

Overall all the methods used in this study (Bracket Penalty method, conjugate direction method, bounding phase method, secant method) combined gives effectives results for smooth constraint optimization problem.

REFERENCES

[1] Kalyanmoy Deb, Optimization for Engineering Design—Algorithms and Examples, Second Edition, 2012.

[2] Singiresu S. Rao, Optimization : Theory and applications, 4th edition,

Appendix: Constraint Optimization Problems

Problem 1:

$$\begin{aligned} \min \quad & f(x) = (x1 - 10)^3 + (x2 - 20)^3, \\ \text{subject to, } & g1(x) = (x1 - 5)^2 + (x2 - 5)^2 - 100 \geq 0, \; g2(x) \\ & = 82.81 - (x1 - 5)^2 - (x2 - 5)^2 \leq 0, \\ & 13 \leq x1 \leq 20, \quad 0 \leq x2 \leq 4. \end{aligned}$$

- Number of variables: 2 variables.
- The global minima: x* = (14.095,0.842), f(x*) = -6961.813

Problem 2:

$$\text{Max } f(x) = \frac{(\sin 2\pi x_1)(\sin 2\pi x_2)}{x_1^3(x_1+x_2)}$$

$$\text{Subject to, } g_1(x) = x_1^2 - x_2 + 1 \leq 0,$$

$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0,$$

$$0 \leq x_1 \leq 10, \qquad 0 \leq x_2 \leq 10.$$

- Number of variables: 2 variables.
- The global minima: $x^* = (1.227, 4.245)$, $f(x^*) = 0.0958$.

Problem 3:

$$\min f(x) = x_1 + x_2 + x_3$$

$$\text{subject to, } g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0,$$

$$g_2(x) = -1 + 0.0025(-x_4 + x_5 + x_7) \leq 0, \; g_3(x) = -1$$

$$+ 0.01(-x_6 + x_8) \leq 0,$$

$$g_4(x) = 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \leq 0,$$

$$g_5(x) = x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \leq 0, \; g_6(x) = x_3x_5 -$$

$$x_3x_8 - 2500x_5 + 1250000 \leq 0, \; 100 \leq x_1 \leq 10000$$

$$1000 \leq x_i \leq 10000, \; i = 2, 3$$

$$10 \leq x_i \leq 1000, \; i = 4, 5, \ldots, 8$$

- Number of variables: 8 variables.
- The global minima: x* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162,395.5979), f (x*) = 7049.3307.