SRC TR 85-51

An Approach for Building a Rule-Based System for Design Optimization

by

S. Azarm and M. Pecht

An Approach for Building a Rule-Based System for Design Optimization

by

Shapour Azarm[1] and Michael Pecht[2]

Department of Mechanical Engineering[3]
The University of Maryland
College Park, MD 20742

[1]The work of this author was supported partially through the General Research Board Grant, and partially through NSF Grant NSFD CDR-85-00108.

[2]The work of this author was supported partially through NSF Grant NSFD-CDR-85-00108.

[3]Assistance in test results was provided by R.E. Lotkowictz and G. Ravi.

Abstract-

A strategy for design optimization of nonlinearly constrained problems is presented. The strategy combines techniques used in production rule systems with an optimization procedure dealing with local monotonicity and sequential quadratic programming techniques. The rule system is based on observations obtained by applying the optimization procedure to different classes of test problems. The observations made may be incorporated in a rule-based system in such a way that if its premise is true, then the action part of the rule is concluded. This is the first step at developing such a rule-based system for design optimization.

1. Introduction

Traditionally the design process has been based on trial-and-error methods, with high quality designs achieved only as experience has accumulated. However, whenever it has been possible to specify an exact or even an approximate relationship between the properties of the object being designed and the design variables, then the use of an optimization technique has enabled the quick generation of a good design. Optimization techniques are most appropriate when there exists a comparatively large number of possible values for the design variables which would all produce a feasible solution. The optimization task is then to find the best acceptable design among all possible designs. Here, the general form of the problem statement to be discussed is given:

$$\begin{aligned} &\text{minimize } f(\underset{\sim}{x}) \\ &\text{subject to} \qquad (1) \\ &\qquad g_j(\underset{\sim}{x}) = 0 \qquad j = 1, \ldots, m \\ &\qquad g_j(\underset{\sim}{x}) \leq 0 \qquad j = (m+1), \ldots, p \end{aligned}$$

where f and g_j are (scalar) objective and constraint functions, and $\underset{\sim}{x}$ is a n-vector of design variables. It is assumed that at least one of the functions (objective or constraints) is nonlinear, i.e. a nonlinear programming (NLP) problem. Solution strategies for the NLP problem have maintained a high degree of interest on the part of both engineers and operation researchers [1-4]. These strategies generally utilize techniques which are based on local information. For example, in one class of gradient-based techniques, the partial derivatives of the objective and constraint functions at the current

point are calculated (local information), and then the direction of minus the gradient of the objective function (or the reduced gradient of the objective function, if there is any equality constraint) is used to obtain the next point while satisfying the constraints. The information provided by the gradient is based on the linearization of the functions at the current point (local information), and may not describe the behavior of the functions involved properly (incomplete information). Therefore, while the algorithm adequately solves one class of problems, it may perform poorly on others, unless extensive "tuning" on the algorithm is done by the user. This has led to a general feeling in the industry that only experts in design optimizaton can apply optimization techniques.

In this paper, we present an approach to the solution of nonlinear programming problems which goes beyond the traditional optimization techniques. The approach couples optimization techniques with the observations made from the test results of different classes of NLP problems. The optimization technique used is based on a local monotonicity analysis [5,6] combined with a sequential quadratic programming technique [7]. A short description of the program, its major features, and use of rule-based techniques to enhance the program procedures are presented. Various test problems and results are then examined to show the effectiveness and variability of design strategies, and to identify areas where additional knowledge and subsequent rules can be meaningfully employed.

2. Description of the Program

The optimization program described here is an extension of the one explained in [6]. It is based on the observation [8-10] that in design optimization,

there usually exists a large number of inequality constraints, many of them satisfied as equalities at the optimum (active constraint). The program utilizes an active set strategy based on local monotonicity information. Two rules used in local monotonicity analysis [5] are repeated here for convenience. Referring to problem (1) we have the following rules:

(1). If the objective function is monotonic with respect to (w.r.t.) a particular variable in the neighborhood of a local minimum, then there exists at least one active constraint with opposite monotonicity w.r.t. that variable in that neighborhood.

(2). If the objective function is stationary w.r.t. a particular variable in the neighborhood of a local minimum, then either all constraints containing that variable are inactive, or there exists at least two active constraints having opposite monotonicity w.r.t. that variable in that neighborhood.

The rules can be viewed as a special case of the Karash-Kuhn-Tucker (KKT) optimality conditions [5]. Since both rules identify the candidate active constraints, a selection criterion is necessary. The selection criterion (which is also in the form of rules) utilizes a local dominance criterion to select the active constraint per rule in a given iteration. If the local prediction of monotonicity is untrue, corrective action is taken, such as a line search between the points generated by two consecutive iterations. We summarize here the basic steps of the algorithm:

Given an initial point as the current point $\bar{x}$;

Step 1:

Find partial derivatives of the objective and constraint functions. If there are some constraints active at this point, then the partial constrained derivatives are evaluated [11].

Step 2:

If $||\nabla f(\bar{\underset{\sim}{x}})|| < \varepsilon$, and if

(a) $\bar{\underset{\sim}{x}}$ is feasible, then check KKT optimality conditions, if they are satisfied, then $\bar{\underset{\sim}{x}} = \underset{\sim}{x}^*$ and stop; otherwise deactivate constraint(s) with negative Lagrange multiplier(s) and go to step 1:

(b) $\bar{\underset{\sim}{x}}$ is infeasible, then deactivate the current active set and go to Step 1:

Otherwise, continue to Step 3.

Step 3:

In the objective function, select the variable (to be referred as the active variable) for which the objective function has the largest absolute partial derivative, continue to Step 4.

Step 4:

Apply first and second monotonicity rules to identify the active constraint(s). If no constraint is active, go to Step 5; otherwise go to Step 6.

Step 5:

Move along a descent direction to a new point and then go to Step 1.

Step 6:

If estimated monotonicities are preserved, go to Step 1. Otherwise deactivate the constraints associated with offending monotonicities. If monotonicity estimates generate violations pertaining to the objective function, do a one-dimensional search. If the violations pertain to the constraints, make a descent move. Then return to Step 1.

The preceding algorithm has been executed for a number of design and test problems [6,10]. The results suggested that further improvement in the algorithm was possible. In particular, to improve the reliability of the algorithm, a sequential quadratic programming (SQP) technique similar to that suggested by Powell [7] was introduced into the program. Transition from a local monotonicity strategy to the sequential quadratic programming technique occurs whenever there is no improvement in the objective function value after a specified number of iterations. The sequential quadratic programming solves a quadratic programming subproblem in each iteration. This subproblem is an approximation of the Lagrangian subject to linearized constraints of (1), and it is guaranteed to have a positive definite Hessian. The subproblem is stated in the following form:

$$\text{minimize} \qquad Q(\underset{\sim}{\delta}) = f(\bar{\underset{\sim}{x}}) + \underset{\sim}{\delta}^t \nabla f(\bar{\underset{\sim}{x}}) + (1/2)\underset{\sim}{\delta}^t B(\bar{\underset{\sim}{x}}, \bar{\underset{\sim}{\lambda}})\underset{\sim}{\delta}$$

$$\text{subject to} \qquad (2)$$

$$\nabla g_j^t(\bar{\underset{\sim}{x}})\underset{\sim}{\delta} + g_j(\bar{\underset{\sim}{x}}) = 0 \qquad j = 1,\ldots,m$$

$$\nabla g_j^t(\bar{\underset{\sim}{x}})\underset{\sim}{\delta} + g_j(\bar{\underset{\sim}{x}}) \leq 0 \qquad j = (m+1),\ldots,p$$

where

$$\underset{\sim}{\delta} = \underset{\sim}{x} - \overline{\underset{\sim}{x}}$$

$$B \simeq \nabla_{xx} L\,(\overline{\underset{\sim}{x}},\ \overline{\underset{\sim}{\lambda}})$$

$$L(\overline{\underset{\sim}{x}},\underset{\sim}{\lambda}) = f(\overline{\underset{\sim}{x}}) + \sum_{j=1}^{p} \lambda_j g_j(\overline{\underset{\sim}{x}}) \qquad (3)$$

The solution of this quadratic programming subproblem estimates the Lagrange multipliers and direction of search $\underset{\sim}{\lambda}^j$, δ^j used in a subsequent one-dimensional search. This one-dimensional minimization has two goals: to decrease the objective function and to minimize the constraint infeasibilities. The function used for one-dimensional minimization is:

$$\phi(\alpha) = f(\underset{\sim}{x}) + \sum_{j=1}^{m} u_j\ |g_j(\underset{\sim}{x})| + \sum_{j=m+1}^{p} u_j |\min(0, g_j(\underset{\sim}{x}))|$$

where

$$\underset{\sim}{x} = \overline{\underset{\sim}{x}} + \alpha\underset{\sim}{\delta}^i \qquad \text{and } u_j \geq 0$$

Here, we select $u_j = |\lambda_j^1|$ for the first iteration and

$$u_j = \max[|\lambda_j^i|,\ 1/2\ (u_j^{i-1} + |\lambda_j^i|)]$$

for subsequent iterations to guarantee convergence [7]. The program is written in FORTRAN and implemented on an IBM-AT microcomputer.

3. Knowledge-Based Optimization Program

The need for developing optimization programs based on information other than that used by the traditional nonlinear programming techniques was discussed previously in [5,10,13]. In that research the idea was to incorporate available global knowledge for a particular problem, with nonlinear programming techniques. In most cases, the knowledge can be organized in the form of rules describing possible constraint activity or inactivity, redundancy, and dominance. In Li and Papalambros [13] it was proposed that rules be organized in a production system that made deductions about possible active constraints in the problem.

Here the idea is to use different local optimization strategies and observe the effect of each strategy on the overall performance of the NLP method using a set of test problems. The observations made here form the premise of a production rule system. If the premise is true, then the action part of the rule is concluded:

if premise then action

The production rule system constructed this way may not be deterministic. In that case, based on the degree of certainty in the premise, the strength of the action is modified [14]. Such a rule-based system is particularly useful in nonlinear programming methods because there is no single NLP method which can solve all classes of nonlinear problems unless extensive "tuning" in the various parameters is done within the program.

One particular requirement of an intelligent optimization analysis is the determination of a good initial strategy which can be altered in the midst of

the program execution. This in turn requires that the selection process examine all possible states as well as the history of the process.

There are various types of selection processes, each of which can change the behavior and convergence of the optimization algorithm. This will be described in the section of test results. The choice of a selection process is dependent on the class of the optimization problem.

In this research, the size of the selection space which will generate the proper solution is unknown. For this reason, we attempted to identify all possible states, with checks to see if rules could be discarded or combined, in an effort to collapse the selection space. For a large selection space this tends to be difficult. However, as constraints are placed on the structure, the number of possible selection processes becomes reduced. It has been noted that there reaches a point where selection rules aid in simplifying the analysis process and reducing the execution time. Presently, we have examined the applicability of a production rule scheme based on identifying classes of solution paths. In particular, the selection of an "active variable" as an initial strategy parameter, can be based on a variety of factors pertaining to the partial derivative of the objective function w.r.t. that variable.

The termination method involves examining the parameters which measure the degree of success. They include: (1) the number of objective function evaluations; (2) the number of constraint functions evaluations and (3) number of gradient evaluations.

4. Test Problems

To make observations with regard to the performance of the program described in section 2, a set of test problems have been selected from Hock and

Schittkowski [12]. Since the test problems have different structure, a classification number is defined. Following the practice of Hock and Schittkowski with a slightly different notation, we define the sequence of letters: OCS-N. The following list gives all possible abbreviations which could replace the letters O, C, S, and N for the tested problems:

O : Information about the Objective function

O=L: Linear objective function

O=Q: Quadratic objective function

O=P: Generalized Polynomial objective function

C : Information about the constraint functions

C=L: Linear constraint functions

C=Q: Quadratic constraint functions

S : Information about the Starting point

S=F: Feasible starting point

S=I: Infeasible starting point

N : Problem number in Hock & Schittkowski [12]

As an example, consider the following NLP problem:

minimize $f(x) = x_1 x_2 x_3$

subject to $x_1^2 + 2x_2^2 + 4x_3^2 - 48 \leq 0$

and with the starting point: $x = (1,1,1)^t$

This problem is classified as PQF-29 since the objective function is Polynomial, the constraint is Quadratic, the starting point is Feasible, and it is problem No. 29 in [12].

We now summarize the abbreviations used in Table 1 to describe the test problems:

TP : Test problem number

OCS-N : Classification of the test problem

NV : Number of variables

NEQ : Number of equality constraints

NC : Total number of constraints, i.e., equalities and inequalities

NACTC : Total number of active constraints

$f(x_*)$: Objective function value at the optimal solution

The test problems considered in this study have 2 to 15 variables with 1 to 22 constraints. In 17 of the test problems, there are as many variables as there are active constraints at the optimum.

5. Test Results

The numerical results of the program testing are listed in Table 2. The abbreviations used in the table are described here:

TP : Test problem number

SU : strategy used:

SU=LA; select the active variable for which the objective function has the Largest Absolute partial derivative.

SU=LN; select the active variable for which the objective function has the Largest Negative partial derivative.

SU=LP; select the active variable for which the objective function has the Largest Positive partial derivative.

NF : Number of objective evaluations

NG : Number of constraint functions evaluations

NDF : Number of gradient evaluations of objective function

NDG : Number of gradient evaluations of constraint functions

We have selected the most efficient strategy for a problem, to be the one with the lowest value of the TOTAL, i.e.:

$$TOTAL = NF + NG + NDF(\text{or } NDG)$$

If one strategy had the lowest TOTAL, it was assigned a probability of one. If two strategies had identical lowest totals, then each of those strategies was assigned a probability of one-half. Finally, if each strategy had the same TOTAL, then each was assigned a probability of one-third.

Once this information was gathered for all the problems executed with all the strategies, an analysis was done to detrmine what the best strategy or combination of strategies is to solve an NLP problem of a particular class. The following simple probability calculation was done for each class of NLP problems:

$$Pr(LX) = \frac{\Sigma\ Pr\ (LX/PROBLEM)}{NPC}$$

where,

1. Pr(LX) = Probability of success of strategy LX for that class.
2. Pr(LX/PROBLEM) = Probability of success of a strategy for a problem.
3. NPC = Number of problems in the particular class of problems being analyzed.

Thus, the strategy that had the greatest Pr(LX) was most likely the best strategy to solve that class of problems. The results of our efforts are shown in Table 3. The table includes the probability of success of a strategy for a particular class of NLP problems. The strategy with the highest probability is most likely the best strategy to solve that class of problems. Table 4 shows the classes of problems and the best strategy for each class.

In addition to indicating the best strategy for each class, Table 4 also suggests a global strategy based on the feasibility of the starting point. If the starting point were infeasible, the best strategy to solve the NLP problem is LN. On the other hand, if the starting point is feasible, the optimal strategy in LP, except for PLF class. In essence this shows perhaps that the structure of the objective function and the constraints is not as important as the feasibility of the starting point in determining the appropriate strategy.

6. Conclusion

The main thrust of this paper is to emphasize the need and feasibility of developing an optimization program which uses knowledge other than that traditionally used in optimization strategies. This knowledge is based on the

observations which are drawn by applying the optimization program on different classes of test problems. The observations may then be used to develop a rule-based system which determines a course of action based on the results of the applied rules.

TABLE 1: List of Test Problems

TP	OCS-N	NV	NC	NACTC	$f(x_*)$
1	PPF-93	6	8	2	135.1
2	PPF-100	7	4	2	680.6
3	PPF-26	3	1	1	0
4	PPI-101	7	20	3	1810
5	PPI-102	7	20	3	911.9
6	PPI-71	4	10	3	17.01
7	LQI-10	2	1	1	1.
8	LQI-95	6	16	6	0.0156
9	LQI-96	6	16	6	0.0156
10	LQI-97	6	16	6	3.136
11	LQI-98	6	16	6	3.136
12	LQI-106	8	22	6	7049
13	PQI-15	2	3	2	306.5
14	PQI-16	2	5	1	0.25
15	PQI-17	2	5	2	1
16	PQI-19	2	6	2	-6962
17	PQI-20	2	5	2	38.2
18	PQI-27	3	1	1	4
19	PQF-29	3	1	1	-22.63
20	PQF-33	3	6	3	-4.586
21	PQF-117	15	20	11	32.35
22	PLF-24	2	5	2	-1
23	PLF-36	3	7	3	-3300
24	PLF-37	3	8	1	-3456
25	PLF-86	5	15	4	32.35
26	QLF-35	3	4	1	0.1111
27	QLF-44	4	10	4	-15
28	QLF-76	4	7	2	-4.68
29	QLF-74	4	13	3	5126
30	QLF-75	4	13	4	5174
31	QQF-30	3	7	2	1
32	QQF-31	3	7	1	6
33	QQF-43	4	3	2	-44
34	QQF-84	5	16	5	-0.528×10^7
35	QQF-113	10	8	6	24.31
36	QQF-12	2	1	1	-30
37	QQI-14	2	2	2	1.39
38	QQI-18	2	6	1	5
39	QQI-22	2	2	2	1
40	QQI-23	2	9	2	2
41	QQI-65	3	7	1	0.95
42	QQI-83	5	16	5	-0.3067×10^5

TABLE 2: Test Results

TP	SU	NF	NG	NDF or NDG
1	LA	141	141	20
	LN	141	141	20
	LP	141	141	20
2	LA	1268	3118	144
	LN	283	684	21
	LP	142	378	13
3	LA	220	850	55
	LN	133	258	34
	LP	133	258	34
4	LA	1172	1172	104
	LN	1172	1172	104
	LP	1172	1172	104
5	LA	1212	1212	109
	LN	1212	1212	109
	LP	1212	1212	109
6	LA	238	238	13
	LN	238	238	13
	LP	226	226	11
7	LA	403	797	22
	LN	403	797	22
	LP	403	797	22
8	LA	479	479	10
	LN	479	479	10
	LP	479	479	10
9	LA	479	479	10
	LN	479	479	10
	LP	479	479	10
10	LA	570	570	15
	LN	570	570	15
	LP	570	570	15
11	LA	570	570	15
	LN	570	570	15
	LP	570	570	15
12	LA	1153	1153	44
	LN	1153	1153	44
	LP	1153	1153	44

13	LA	19	26	6
	LN	85	90	14
	LP	42	54	13
14	LA	87	87	14
	LN	87	87	14
	LP	87	87	14
15	LA	"Could not find feasible point"		
	LN	109	120	22
	LP	"Could not find feasible point"		
16	LA	91	124	18
	LN	45	53	16
	LP	91	124	18
17	LA	126	126	27
	LN	126	126	27
	LP	118	118	25
18	LA	254	245	17
	LN	254	245	17
	LP	254	245	17
19	LA	432	921	108
	LN	432	921	108
	LP	97	134	25
20	LA	26	34	6
	LN	46	47	12
	LP	26	34	6
21	LA	337	337	21
	LN	337	337	21
	LP	337	337	21
22	LA	19	26	6
	LN	19	26	6
	LP	40	43	6
23	LA	34	46	8
	LN	34	46	8
	LP	45	48	12
24	LA	63	63	15
	LN	58	58	15
	LP	63	63	15
25	LA	85	85	14
	LN	85	85	14
	LP	85	85	14

26	LA	34	34	8
	LN	58	58	15
	LP	34	34	8
27	LA	79	92	15
	LN	74	86	15
	LP	68	69	14
28	LA	180	542	16
	LN	162	252	22
	LP	193	280	21
29	LA	382	382	19
	LN	382	382	19
	LP	616	616	15
30	LA	317	317	18
	LN	317	317	18
	LP	491	491	14
31	LA	98	178	10
	LN	66	67	17
	LP	98	178	10
32	LA	187	423	13
	LN	142	409	14
	LP	170	262	23
33	LA	987	1957	177
	LN	399	2780	32
	LP	244	510	31
34	LA	73	73	12
	LN	73	73	12
	LP	73	73	12
35	LA	265	265	24
	LN	265	265	24
	LP	265	265	24
36	LA	83	237	24
	LN	83	237	24
	LP	149	311	28
37	LA	166	166	9
	LN	166	166	9
	LP	166	166	9
38	LA	173	578	19
	LN	45	48	16
	LP	173	578	19

39	LA	91	96	16
	LN	91	96	16
	LP	91	96	16
40	LA	19	29	6
	LN	47	57	16
	LP	19	29	6
41	LA	79	79	12
	LN	108	108	20
	LP	108	108	20
42	LA	207	207	12
	LN	207	207	12
	LP	207	207	12

TABLE 3: Strategy Effectiveness

CLASS (NPC)	STRATEGY	PROBABILITY OF SUCCESS
PPF(3)	LA	0.11
	LN	0.28
	LP	0.61
PPI(3)	LA	0.33
	LN	0.33
	LP	0.33
LQI(6)	LA	0.33
	LN	0.33
	LP	0.33
PQI(6)	LA	0.28
	LN	0.44
	LP	0.28
PQF(3)	LA	0.28
	LN	0.11
	LP	0.61
PLF(4)	LA	0.33
	LN	0.58
	LP	0.09
PGI(3)	LA	0.50
	LN	0.50
	LP	0.00
QLF(2)	LA	0.17
	LN	0.33
	LP	0.50
QQF(6)	LA	0.19
	LN	0.36
	LP	0.45
QQI(6)	LA	0.36
	LN	0.45
	LP	0.19

TABLE 4: Best Strategies

Class (NPC)	Strategy
PPF(3)	LP
PPI(3)	LA,LN, or LP
LQI(6)	LA,LN, or LP
PQI(6)	LN
PQF(3)	LP
PLF(4)	LN
PGI(3)	LA or LN
QLF(2)	LP
QQF(6)	LP
QQI(6)	LN

7. References

1. G. V. Reklaitis, et al., Engineering Optimization, Wiley, New York, 1983.

2. J. N. Siddall, Optimal Engineering Design, Marcel Dekker, New York, 1982.

3. M. W. Bazaraa, and Shetty, C.M., Nonlinear Programming, Wiley, 1979.

4. G. P. McCormick, Nonlinear Programming, Wiley, 1983.

5. S. Azarm and P. Papalambros, "A Case for a Knowledge-Based Active Set Strategy," ASME Trans., Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 106, No. 1, March 1984.

6. S. Azarm and P. Papalambros, "An Automated Procedure for Local Monotonicity Analysis," ASME Trans., Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 106, No. 1, March 1984.

7. M.J.D. Powell, "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," Proceedings of the 1977 Dundee Conference on Numerical Analysis, Lecture Notes in Mathematics, Vol. 630, Springer-Verlag, Berlin, pp. 144-157, 1978.

8. D. J. Wilde, Globally Optimal Design, Wiley, 1978.

9. P. Papalambros, "Monotonicity Analysis in Engineering Design Optimization", Ph.D. Dissertation, Department of Mechanical Engi ering, Stanford University, 1979.

10. S. Azarm, Local Monotonicity in Optimal Design, Ph.D Dissertation, Department of Mechanical Engineering and Applied Mechanics, The University of Michigan, Ann Arbor, 1984.

11. D. J. Wilde and C. Beightler, Foundation of Optimization, Prentice-Hall, N.J., 1979.

12. W. Hock and K. Schittkowski, Test Examples for Nonlinear Programming Codes, Springer-Verlag, New York, 1980.

13. H. L. Li, and P. Papalambros, "A Production System for Use of Globally Optimization Knowledge," ASME paper 84-DET-194; Journal of Mechanisms, Transmissions and Automation in Design. (to be published).

14. W. J. Van Melle, System Aids in Constructing Consultation Programs, UMI Research Press, Ann Arbor, Michigan, 1981.