CLIN. CHEM. 39/9, 1972–1978 (1993)

Genetic Algorithms-Based Design and Optimization of Statistical Quality-Control Procedures

Aristides T. Hatjimihail

In general, one cannot use algebraic or enumerative methods to optimize a quality-control (QC) procedure for detecting the total allowable analytical error with a stated probability with the minimum probability for false rejection. Genetic algorithms (GAs) offer an alternative, as they do not require knowledge of the objective function to be optimized and can search through large parameter spaces quickly. To explore the application of GAs in statistical QC, I developed two interactive computer programs based on the deterministic crowding genetic algorithm. Given an analytical process, the program "Optimize" optimizes a user-defined QC procedure, whereas the program "Design" designs a novel optimized QC procedure. The programs search through the parameter space and find the optimal or near-optimal solution. The possible solutions of the optimization problem are evaluated with computer simulation.

Indexing Terms: *computer simulation · deterministic crowding · binary coding · error detection · statistical process monitoring*

Quality-control (QC) procedures are characterized by their probabilities for error detection and for false rejection. An optimum QC procedure should have stated probabilities for detecting the critical errors, with the minimum probability for false rejection. Critical errors are considered the maximum allowable errors, "defined in such a way that an upper bound has been set on the (clinical) type I error" (*1*). The clinical type I error is the probability for rejection of the true hypothesis that there is no significant change in an analyte concentration in a patient. The hypothesis is rejected when the observed change in the analyte concentration is greater than the maximum allowable error.

Single-rule and multirule QC procedures can be designed. Each rule is a statistical decision rule, with one or more parameters. The performance of the QC procedures is described by their power functions (*2*) that relate their probabilities for rejection vs the analytical error. The probabilities for rejection are estimated by computer simulation (*3–7*), because the algebraic definition of the power functions is usually complex. Therefore, we cannot use algebraic methods to optimize QC procedures. Use of enumerative methods would be tedious, especially with multirule procedures, because the points of the parameter space to be searched grow exponentially with the number of parameters to be optimized. Optimization methods based on genetic algorithms (GAs) (*8, 9*) offer an appealing alternative: they are robust search algorithms that do not require knowledge of the objective function and can search through large spaces quickly. GAs have been used successfully to solve a variety of complex optimization problems (*10*), including the control of a gas pipeline system (*11*), the optimization of an aerospace-related control system (*12*), the design and optimization of neural networks (*13, 14*), and the design of a high-by-pass jet engine turbine (*8*).

GAs were derived from processes of molecular biology and the evolution of life. Their operators—crossover, mutation, and reproduction (*15*)—are isomorphic with the synonymous biological processes. Instead of DNA or RNA strands, GAs process strings of symbols of finite length; these symbols encode the parameters of the objective function to be optimized. Usually, binary strings are used, i.e., strings of zeros and ones (Figure 1*A*).

Initially, a population of n randomly defined strings of the desired length is generated. Then, during successive generations, the strings are:

1. Crossed over. The strings are paired randomly and swap homologous substrings (Figure 2*A*). Crossover is an effective means of exchanging information and combining partial solutions (*12*).
2. Mutated. Mutation is the occasional alteration of a string symbol (Figure 2*B*). Mutation reintroduces diversity into the population of the strings.
3. Translated. Translation is used to define the parameters of the objective function (*Appendix I*).

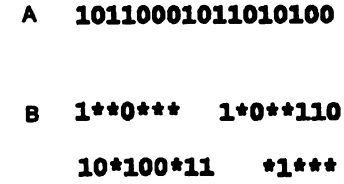

Fig. 1. (*A*) A binary string; (*B*) schemata; asterisks denote variables

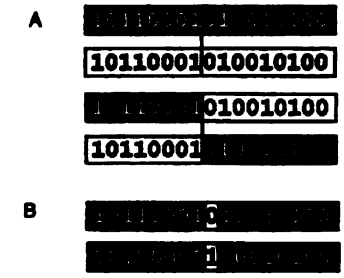

Fig. 2. (*A*) Crossover; (*B*) mutation

Microbiology Laboratory, Health Center of Prosotsane, GR-662 00 Prosotsane, Greece, and Hellenic Complex Systems Laboratory, P.O. Box 56, GR-661 00 Drama, Greece.

Address for correspondence: P.O. Box 56, GR-661 00 Drama, Greece. Fax (30)(521) 36 574.

Received May 12, 1993; accepted June 24, 1993.

4. Reproduced. The corresponding objective function values are used to select the fittest strings to be reproduced for the next generation. Various selection schemes can be applied, including tournament selection (*16*), sharing (*17*), and deterministic crowding (*18*).

The essence of the genetic search is the implicit parallel processing of the schemata (*19*). The schemata are partially defined substrings (Figure 1*B*). The order of a schema is defined by the number of its variables. In a population of size n, approximately n^3 schemata are processed in parallel (*9*). Low-order, short schemata with above-average fitness are called building blocks; these receive exponentially increasing numbers in succeeding generations, recombining thereafter to form optimal or near-optimal solutions (*9*).

To explore the application of the GAs in statistical quality control, I have developed the two following interactive microcomputer programs based on GAs:

1. "Optimize," for the optimization of statistical QC procedures. The user: (*a*) defines a QC procedure as a Boolean proposition, using any of 10 generic QC rules (*Appendix IV*); (*b*) selects the parameters to be optimized and defines the rest of them; and (*c*) defines the parameters of the analytical process and of the genetic search. The program searches through the parameter space and finds an optimal or near-optimal solution.

2. "Design," for the design of optimized QC procedures. The user defines the number of QC rules of the procedure and the parameters of the analytical process. Then the program defines an optimized QC procedure, using any of four generic QC rules (*Appendix IV*).

Materials and Methods

The Program "Optimize"

The program is written in Pascal (*20, 21*) for an IBM or compatible microcomputer, with an Intel 80486 central processing unit under MS-DOS, version 6.0. It includes the following units:

Rules unit. The user defines the decision rules by defining up to eight parameters of one or more generic rules and by selecting the parameters to be optimized.

The rules (see *Appendix IV*) can be divided into six classes: single-value rules (*21, 22*), sum rules (*23, 24*), range rules (*23, 24*), mean rules (*25, 26*), standard deviation rules (*23*), and trend rules (*26*).

The rules can be applied across runs, within a run, across levels, or within each level of the controls (*27*).

The user selects the parameters to be optimized and defines the rest of them. The upper and lower bounds of the parameters to be optimized are user-defined.

Interpreter unit. The QC multirule procedures are defined as Boolean expressions composed from:

1. Operands: symbols of the user-defined rules or multirule procedures.

2. Operators: symbols of the logical operations AND, OR, NOT, XOR (exclusive or).

3. Parentheses: as required.

The interpreter accepts those expressions as input.

Random normal deviates generator unit (7). The program assumes a normal distribution of error. The unit generates Box–Muller series (*28*) of random normal deviates from pseudorandom numbers. The pseudorandom number generator is the function "Random" of the compiler.

Genetic algorithms unit. The length k of each substring and the upper and the lower bounds of the parameters are user-defined (see equation 3 of *Appendix I*). The user defines the population n of the strings and also defines the probability P_m that a symbol will be mutated and the probability P_c that a pair of strings will be crossed over during one generation, usually $0 \leq P_m \leq 0.001$ and $0.75 \leq P_c \leq 1$.

Initially a population of n random strings is generated. The length of the strings is

$$l = \sum_{m=1}^{p} k_m$$

where k_m is the length of the substring coding the mth parameter, and p is the number of parameters to be optimized. The fitness f of the translated strings is calculated from the objective function:

$$f = \sqrt{w_{re}(P_{re} - P_{re_c})^2 + w_{se}(P_{se} - P_{se_c})^2 + w_{fr}P_{fr}^2} \quad (1)$$

where P_{re} and P_{se} are the probabilities for random and systematic critical errors detection, P_{fr} is the probability for false rejection of the QC procedure defined by the translated string, P_{re_c} and P_{se_c} are the stated probabilities for critical random and systematic errors detection, and w_{re}, w_{se}, and w_{fr} are user-defined weighing factors. f is minimized.

The strings are evaluated and then reproduced according to the deterministic crowding algorithm (*Appendix II*). The genetic search continues either for 100 generations or until the population of the strings becomes homogenous. The population is considered homogenous when

$$f_m - f_{min} \leq 0.001 \cdot f_{min}$$

where f_m and f_{min} are the mean fitness and the minimum fitness of the strings of the population, respectively.

Simulator unit (7). The user defines the number and the levels (up to three) of the controls. The program simulates 1000 control measurements at each level, in consecutive runs. If n is the number of control measurements at each level of a run, then the number of the simulated runs is $1000/n$.

To estimate the probability for critical random and systematic error detection, the program introduces into the simulated control measurements the critical random and systematic error (*Appendix II*).

The program simulates the following situations:

1. There is error in all the measurements to which the QC procedure is applied.

2. The error is introduced between two consecutive runs.
3. The error is introduced within a run.

The Program "Design"

The units of this program are similar to the units of the previous program, with the following differences:

Rules unit. The rules used by the program (see *Appendix IV*) are single-value rules, range rules, mean rules, and standard deviation rules. The rules are applied across runs and across levels of the controls.

Interpreter unit. The user defines the number q of the rules of the QC procedure. The procedure to be designed by the program can be denoted[1] as:

$$Q_1(n_1, max_1) \# Q_2(n_2, max_2) \# \ldots \# Q_q(n_q, max_q)$$

The priority of the operators is optimized by the program; therefore, the designed procedure may have parentheses.

Random normal deviates generator unit. The pseudorandom number generator is based on the FORTRAN code of Marse and Roberts (*29, 30*).

Genetic algorithms unit. Each QC rule and the respective operator is coded as a substring of 13 bits. Two bits code the class of the rule, two bits code the sample size n, and six bits code the decision limit *max*. One bit codes the kind of operator (AND or OR) and two bits code the priority of the operation. The bounds of the parameters to be optimized (*Appendix IV*) and the parameters of the genetic search are predefined, except the population n of the strings, which is user-defined. The weighing factors w_{re}, w_{se}, w_{fr} of the objective function (equation 1) are set to 1, the probability for crossover to 1, and the probability for mutation to 0.

The genetic search continues for 50 generations.

Simulator unit. The program simulates 6000 control measurements at each level, in consecutive runs.

Application of the Programs

To demonstrate the process of the design and the optimization of a QC procedure, it is assumed that the SD and the bias of an analytical method for the measurement of an analyte are 27 and 10 mg/L, respectively, while the maximum allowable analytical error is 200 mg/L. The upper bound of the clinical type I error is set to 0.01. The stated probabilities for critical random and systematic errors detection are set to 0.6 and 1.0, respectively. Ignoring the preanalytical and biological variation and assuming that we analyze only one sample per test, the critical random and systematic errors calculated from equations 4 to 11 of *Appendix II* are 2.852 and 4.711, respectively. The procedures are tested with one control per level, two levels of controls, no rounding, and between-run error set to zero. It is assumed that the error is introduced between two consecutive runs.

Application of the program "Optimize". The QC procedure to be optimized was defined as:

$$S(1,x) \text{ OR } M(2,y) \text{ OR } D(2,z)$$

The rules were applied across runs and levels of controls.

The length of the substrings that code each of the parameters x, y, and z was set to 9. The lower and the upper bounds of the parameters were 0 and 5.11, respectively. Since the substrings were interpreted as 9-bit binary numbers, the accuracy of the calculation of each parameter was $5.11/(2^9 - 1) = 0.01$. The three substrings were concatenated to a string of length 27. Therefore, the search space included $2^{27} = 134\,217\,728$ points. The initial population n of the strings was set to 600. The weighing factors w_{re}, w_{se}, w_{fr} of the objective function (equation 1) were set to 1, the probability for crossover to 1, and the probability for mutation to 0.

The optimized QC procedure was compared with 48 alternative, commonly used QC procedures[2], including the following, according to Westgard's notation (*31*):

1. The Westgard procedure (*27*).
2. 1_{V_s}, where $V = 2.0 + 0.1k$, $k = 0, 1, 2, \ldots, 20$.
3. $1_{3.0_s}/2_{2.0_s}/R_{4.0_s}$.

To compare the designed QC procedures with the alternative ones, I used a version of the previously described simulation program (*7*), with the simulator unit of the program "Optimize". I performed 21 simulation runs for each procedure, using 21 different sets of series of simulated control measurements. The probabilities for critical errors detection and for false rejection were estimated. For the comparison of the procedures I used a function similar to that of equation 1, i.e.:

$$f_1 = \sqrt{\Delta P_{re}^2 + \Delta P_{se}^2 + P_{fr}^2} \qquad (2)$$

where $\Delta P_{re} = P_{re} - 0.6$, if $P_{re} < 0.6$; otherwise, $\Delta P_{re} = 0$; and $\Delta P_{se} = P_{se} - 1$. The minimum f_1 is considered optimum.

Application of the program "Design". A three-rule QC procedure was designed. The procedure was coded by a string of 39 bits. Therefore the parameter space included $2^{39} = 549\,755\,813\,888$ points. The population n of the strings was set to 200.

The designed QC procedures were compared with the QC procedures of the library, as described in the previous section, using a version of the same simulation program (*7*), with the pseudorandom number generator of Marse and Roberts (*29*).

[1] The following notation is used: Let n be the sample size, i.e., the number of measurements the QC rule is applied upon, and $x(SD)$ the decision limit. Then $S(n, x)$ denotes a single-value rule, $R(n, x)$ a range rule, $M(n, x)$ a mean rule, $D(n, x)$ a standard deviation rule, and $Q_i(n, x)$ any rule from the above. A rule is true if the respective statistic is $>x$. # denotes either the operator AND or the operator OR.

[2] The list of the 48 QC procedures is available from the author.

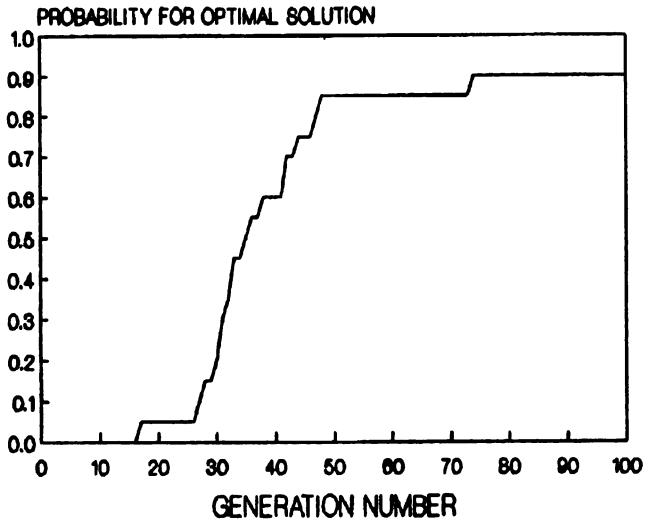

Fig. 3. Cumulative probability distribution of at least one optimal solution vs the generation number, after 20 simulation runs (program "Optimize")

Results

Application of the Program "Optimize"

One optimal solution of the optimization problem was found in 18 runs. The minimum estimated fitness value was $f = 0.0100$ for $x = 2.65$, $y = 2.27$, and $z = 3.40$. The estimated probabilities for critical random and systematic error detection of the so-defined optimal QC procedure were 0.599 and 1.000, respectively, while the estimated probability for false rejection was 0.010.

Figure 3 shows the cumulative probability distribution of finding at least one optimal solution vs the generation number.

I have proven that the program found the only optimal solution in the parameter space of the >130 million points.

Tables 1 and 2 present the mean and SD of the estimated probabilities for critical errors detection and for false rejection, and the values of the function f_1 of the optimized procedure and of the two best procedures of the library.

Application of the Program "Design"

These are two of the procedures, designed by the program, during five consecutive runs, and the respective probabilities for critical errors detection and for false rejection:

1. S(1, 2.7) OR S(2, 2.1) OR S(3, 1.8)
   $P_{re_c} = 0.598$, $P_{se_c} = 0.999$, $P_{fr} = 0.0170$
2. [S(1, 2.7) OR R(2, 3.8)] AND S(1, 2.4)
   $\Leftrightarrow$ S(1, 2.7) OR [R(2, 3.8) AND S(1, 2.4)]
   $P_{re_c} = 0.595$, $P_{se_c} = 0.999$, $P_{fr} = 0.0172$

The procedures were designed after searching $<2 \times 10^{-8}$ of the parameter space.

Tables 3 and 4 present the mean and the SD of the estimated probabilities for critical errors detection and for false rejection, and the values of the function f_1 of the two previous procedures and of the two best procedures of the library.

Table 1. Probabilities for Critical Random (P_{re}) and Systematic (P_{se}) Error Detection, and for False Rejection (P_{fr}), Estimated after 21 Simulation Runs Using the Simulator Unit of the Program "Optimize"

		P_{re}	P_{se}	P_{fr}
S(1, 2.65) OR M(2, 2.27) OR D(2, 340)[a]	Mean	0.5951	0.9998	0.0180
	SD	0.0102	0.0005	0.0042
$1_{3.0s}/(2 \text{ of } 3)_{2.0s}/R_{4s}$[b]	Mean	0.5921	0.9989	0.0180
	SD	0.0071	0.0003	0.0036
$1_{2.6s}$[b]	Mean	0.5888	0.9987	0.0181
	SD	0.0102	0.0006	0.0035

[a] QC procedure optimized by the program "Optimize".
[b] QC procedure of the library.

Table 2. Values of Function f_1, Estimated after 21 Simulation Runs Using the Simulator Unit of the Program "Optimize"

		f_1
S(1, 2.65) OR M(2, 2.27) OR D(2, 340)[a]	Mean	0.0206
	SD	0.0053
$1_{3.0s}/(2 \text{ of } 3)_{3.0s}/R_{4s}$[b]	Mean	0.0208
	SD	0.0037
$1_{2.6s}$[b]	Mean	0.0229
	SD	0.0065

[a] QC procedure optimized by the program "Optimize".
[b] QC procedure of the library.

Discussion

Traditionally, QC has been a compromise between the requirements for greater accuracy and precision and for fewer false rejections (*31, 32*). The concept of the medically allowable analytical error (*1, 31, 33*) offered objective criteria for the evaluation of alternative QC procedures. Nevertheless, the design of QC procedures and the definition of their parameters have been based mainly on the insight of the researchers (*24, 26, 27, 31, 34*), who were presumably using a trial-and-error process.

The usefulness of computer simulation for the design and evaluation of alternative QC procedures (*3–7*) and the power and robustness of GAs as optimization tools (*8–15, 35*) are well established. By combining these two

Table 3. Probabilities for Critical Random (P_{re}) and Systematic (P_{se}) Error Detection and for False Rejection (P_{fr}), Estimated after 21 Simulation Runs Using the Random Number Generator of the Program "Design"

		P_{re}	P_{se}	P_{fr}
S(1, 2.7) OR S(2, 2.1) OR S(3, 1.8)[a]	Mean	0.5961	1.0000	0.0167
	SD	0.0100	0.0000	0.0010
[S(1, 2.7) OR R(2, 3.8)] AND S(1, 2.4)[a]	Mean	0.5924	0.9997	0.0177
	SD	0.0084	0.0008	0.0013
$1_{3.0s}/(2 \text{ of } 3)_{2.0s}/R_{4s}$[b]	Mean	0.5880	1.0000	0.0157
	SD	0.0086	0.0000	0.0011
$1_{2.6s}$[b]	Mean	0.5901	0.9999	0.0189
	SD	0.0099	0.0006	0.0016

[a] QC procedure designed by the program "Design".
[b] QC procedure of the library.

Table 4. Values of Function f_1 Estimated after 21 Simulation Runs Using the Random Number Generator of the Program "Design"

		f_1
S(1, 2.7) OR S(2, 2.1) OR S(3, 1.8)[a]	Mean	0.0188
	SD	0.0036
[S(1, 2.7) OR R(2, 3.8)] AND S(1, 2.4)[a]	Mean	0.0205
	SD	0.0040
$1_{3.0s}/(2 \text{ of } 3)_{2.0s}/R_{4s}$[b]	Mean	0.0210
	SD	0.0043
$1_{2.6s}$[b]	Mean	0.0230
	SD	0.0047

[a] QC procedure designed by the program "Design."
[b] QC procedure of the library.

techniques, the program "Optimize" permits the user to design a QC procedure and to optimize its parameters to detect the critical random and systematic error with stated probabilities, while the probability for false rejection is as low as possible. Then the optimized QC procedure can be compared with alternative QC procedures. In our case the optimized QC procedure is as good as the best of the commonly used ones (Tables 1 and 2). For the calculation of the critical random and systematic error, a new, more general system of equations is used (*Appendix II*). The program is very flexible and searches efficiently through very large parameter spaces (Figure 3), using only objective function values. Quoting Goldberg, "GAs use payoff (objective function) information, not derivatives or other auxilliary knowledge" (*9*). Therefore, GA-based programs, such as the program "Optimize", can be used for the optimization of any QC procedure.

Approaches for the selection of QC procedures from libraries of predefined QC procedures have been described that use the concept of the total allowable analytical error (*3, 36, 37*). The program can augment these approaches, as it permits further optimization of the parameters of the QC procedures that are selected.

The program "Design" illustrates the computational innovation that GAs are generating by combining partial solutions. This kind of methodology is strictly inductive compared with other search methods, which are deductive (*12*). Considering our problem, GAs have generated novel QC procedures at least as good as the best of the commonly used ones (Tables 3 and 4).

Further research should be done for the application of other GAs (*38, 39*) in statistical QC, for optimization of the initial population n of the strings (*40*) and of the parameters of the genetic search, and for improvement of the simulators implemented in personal computers (*6*). The availability of new, advanced operating systems and of faster central processing units will permit the design of faster, more reliable simulators. I am working in that direction.

GAs, as well as neural networks, immune networks, and classifier systems, are examples of complex adaptive systems (*19*) that use metaphorically concepts, principles, and processes that explain natural systems. Nature is the most efficient problem solver; therefore, "natural computation," based on complex adaptive systems, can be an excellent problem-solving method in many fields of science and technology.

Although this is probably the first time that GAs have been applied in statistical QC, I conclude that they can be a powerful research tool to explore alternative QC procedures and to design novel and possibly better ones.

I thank David E. Goldberg for his continuing support and encouragement. I also thank Theofanis T. Hatjimihail, for his help during the preparation of this manuscript.

References

1. Linnet K. Choosing quality-control systems to detect maximum clinically allowable analytical errors. Clin Chem 1989;35:284–8.
2. Westgard JO, Groth T. Power functions for statistical control rules. Clin Chem 1979;25:863–9.
3. Groth T, Falk H, Westgard JO. An interactive computer simulation program for the design of statistical control procedures in clinical chemistry. Comput Programs Biomed 1981;13:73–86.
4. Blum AS. Computer evaluation of statistical procedures, and a new quality-control statistical procedure. Clin Chem 1985;31:206–12.
5. Smith FA, Cossitt NL. Simulated comparison of multi-rule protocols for statistical quality control [Abstract]. Clin Chem 1987;33:909.
6. Parvin CA. Estimating the performance characteristics of quality-control procedures when error persists until detection. Clin Chem 1991;37:1720–4.
7. Hatjimihail AT. A tool for the design and evaluation of alternative quality-control procedures. Clin Chem 1992;38:204–10.
8. Holland JH. Genetic algorithms. Sci Am 1992;267:66–72.
9. Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley, 1989:1–57.
10. Goldberg DE, Milman K, Tidd C. Genetic algorithms: a bibliography (IlliGAL Report 92008). Urbana, IL: University of Illinois, Illinois Genetic Algorithms Laboratory, 1992.
11. Goldberg DE. Computer-aided gas pipeline operation using genetic algorithms and rule learning [Ph.D. dissertation]. Ann Arbor, MI: University of Michigan, 1983.
12. Krishnakumar K, Goldberg DE. Genetic algorithms in control system optimization. Portland, OR: AIAA Guidance, Navigation and Control Conference, 1990:1568–77.
13. Kitano H. Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms (Tech. Rep. No. CMU-CMT-92-134). Pittsburgh, PA: Carnegie Mellon University, Center for Machine Translation, 1992.
14. Whitley D, Starkweather T, Bogart C. Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Comput 1990;14:347–61.
15. Goldberg DE. Genetic algorithms as a computational theory of conceptual design. In: Rzevski G, Adey RA, eds. VI. Applications of artificial intelligence in engineering. London: Elsevier Applied Science, 1991:3–16.
16. Goldberg DE, Deb K. A comparative analysis of selection schemes used in genetic algorithms. In: Rawlins GJE, ed. Foundations of genetic algorithms. San Mateo, CA: Morgan Kaufmann Publishers, 1991:69–93.
17. Deb K, Goldberg DE. An investigation of niche and species formation in genetic function optimization. Proc, Third Int Conf on Genetic Algorithms, 1989:42–50.
18. Mahfoud SW. Crowding and preselection revisited. In: Männer R, Manderick B, eds. Parallel problem solving from nature, Vol. 2. Amsterdam: Elsevier Science Publishers, 1992:27–36.
19. Holland JH. Adaptation in natural and artificial systems. Cambridge, MA: MIT Press, 1992:1–19, 66–88.
20. Turbo Pascal, version 7.0, language guide. Scotts Valley, CA: Borland International, 1992.
21. Turbo Pascal, version 7.0, Turbo Vision programming guide. Scotts Valley, CA: Borland International, 1992.
22. Ehrmeyer SS, Laessig RH, Schell K. Use of alternative rules

(other than the 1_{2s}) for evaluating interlaboratory performance data. Clin Chem 1988;34:250–6.
23. Westgard JO, Klee GG. Quality assurance. In: Tietz NW, ed. Textbook of clinical chemistry. Philadelphia: WB Saunders, 1986: 424–58.
24. Westgard JO, Groth T, Aronsson T, de Verdier C-H. Combined Shewhart–Cusum control chart for improved quality control in clinical chemistry. Clin Chem 1977;23:1881–7.
25. Montgomery DC. Introduction to statistical quality control. New York: John Wiley, 1985:171–219.
26. Cembrowski GS, Westgard JO, Eggert AA, Toren EC Jr. Trend detection in control data: optimization and interpretation of Trigg's technique for trend analysis. Clin Chem 1975;21:1396–405.
27. Westgard JO, Barry PL, Hunt MR, Groth T. A multi-rule Shewhart chart for quality control in clinical chemistry. Clin Chem 1981;27:493–501.
28. Box GEP, Muller ME. A note on the generation of random normal deviates. Ann Math Stat 1958;29:610–1.
29. Marse K, Roberts SD. Implementing a portable FORTRAN uniform (0, 1) generator. Simulation 1983;41:135–9.
30. Law AM, Kelton WD. Simulation modeling and analysis. New York: McGraw-Hill, 1991:420–61.
31. Westgard JO, Barry PL. Cost-effective quality control: managing the quality and productivity of analytical processes. Washington, DC: AACC Press, 1986:33–91, 190–91, 195–217.
32. Duncan AJ. Quality control and industrial statistics. Homewood, IL: Irwin, 1986:476–512.
33. Westgard JO, Groth T, de Verdier C-H. Principles for developing improved quality control procedures. Scand J Clin Lab Invest 1984;44(Suppl 172):19–41.
34. Parvin C. Comparing the power of quality-control rules to detect persistent systematic error. Clin Chem 1992;38:358–63.
35. Deb K. Binary and floating-point function optimization using messy genetic algorithms (IlliGAL Report 91004). Urbana, IL: University of Illinois, Illinois Genetic Algorithms Laboratory, 1991:35–146.
36. Westgard JO, Quam EF, Barry PL. Selection grids for planning QC procedures. Clin Lab Sci 1990;3:273–80.
37. Hatjimihail AT. Computer aided selection of alternative quality control procedures [Abstract]. Second Mediterranean Medical Meeting. Athens: Mediterranean Medical Society, 1992:144.
38. Goldberg DE, Korb B, Deb K. Messy genetic algorithms: motivation, analysis, and first results. Complex Systems 1989;3: 493–530.
39. Goldberg DE, Deb K, Horn J. Massive multimodality, deception, and genetic algorithms. In: Männer R, Manderick B, eds. Parallel problem solving from nature, Vol. 2. Amsterdam: Elsevier Science Publishers, 1992:37–46.
40. Goldberg DE, Kalyanmoy D, Clark JH. Genetic algorithms, noise, and the sizing of populations. Complex Systems 1992;6:333–62.
41. Westgard JO, Wiebe DA. Cholesterol operational process specifications for assuring the quality required by CLIA proficiency testing. Clin Chem 1991;37:1938–44.
42. Press WH, Flannery BP, Teukolsky SA, Vetterling WT. Numerical recipes in Pascal. Cambridge: Cambridge University Press, 1989.
43. Fraser CG, Hyltoft Petersen P, Lytken Larsen M. Setting analytical goals for random analytical error in specific clinical monitoring situations. Clin Chem 1990;36:1625–8.

Appendix I. Coding of the Parameters

The parameters to be optimized are coded as binary substrings of zeros and ones, which are easily interpreted as binary numbers. A concatenated, mapped, unsigned binary coding is used (*29*). Therefore, if a substring S of length k, coding the parameter D, is interpreted as the binary number b, then the substring is decoded as

$$D = D_{min} + \frac{b\,(D_{max} - D_{min})}{2^k - 1} \qquad (3)$$

where D_{max} and D_{min} are the upper and lower bounds of D. Consequently, the accuracy of the calculation of the parameter D is

$$a = \frac{D_{max} - D_{min}}{2^k - 1}$$

Appendix II. Calculation of the Critical Random and Systematic Errors

Assuming that $f(z) = 1/\sqrt{2\pi}e^{z^2/2}$ or that the probability density function of z is the unit normal distribution, one can calculate the critical errors from the following equations (*1, 7, 41*):

$$P(z > z_1) + P(z > z_2) = P(|TE| > TE_a) \qquad (4)$$

$$P(z > z_3) + P(z > z_4) = P(|TE| > TE_a) \qquad (5)$$

$$s_{t_c} = \frac{TE_a - |\,b_{spec} + b\,|}{z_1} \qquad (6)$$

$$s_{t_c} = \frac{TE_a + |\,b_{spec} + b\,|}{z_2} \qquad (7)$$

$$s_{t_c}^2 = \frac{s_{sub}^2}{n_{test}} + \frac{s_{spec}^2}{n_{test} \cdot n_{spec}} + \frac{(RE_c \cdot SD)^2}{n_{test} \cdot n_{spec} \cdot n_{samp}} \qquad (8)$$

$$SE_c = \frac{TE_a - |\,b_{spec} + b\,|}{s_t} - z_3 \qquad (9)$$

$$SE_c = -\frac{TE_a + |\,b_{spec} + b\,|}{s_t} + z_4 \qquad (10)$$

$$s_t^2 = \frac{s_{sub}^2}{n_{test}} + \frac{s_{spec}^2}{n_{test} \cdot n_{spec}} + \frac{SD^2}{n_{test} \cdot n_{spec} \cdot n_{samp}} \qquad (11)$$

where SD and b are the standard deviation and bias of the analytical method, TE the total error, TE_a the total allowable analytical error, s_{t_c} the total critical standard deviation, s_t the total standard deviation when $RE = 1$, s_{sub} the standard deviation of the within-subject biological variation, s_{spec} and b_{spec} the standard deviation and bias due to the sampling procedure, n_{test} the number of tests performed, n_{spec} the number of specimens per test, n_{samp} the number of analyzed samples per specimen, RE_c and SE_c the random and systematic critical analytical errors (expressed as multiples of the SD), and z_1, z_2, z_3, z_4 real numbers. The equations can be solved numerically, using Brent's method (*42*). Considering monitoring purposes (*43*), the critical random error can be calculated from the equations:

$$P(z > z_1) = \frac{P(\,|\,TE\,|\, > TE_a)}{2} \qquad (12)$$

$$\sqrt{2} \cdot s_{t_c} = z_1 \cdot TE_a \quad (13)$$

$$s^2_{t_c} = s^2_{sub} + \frac{s^2_{spec}}{n_{spec}} + \frac{(RE_c \cdot SD)^2}{n_{spec} \cdot n_{samp}} \quad (14)$$

Appendix III. The Deterministic Crowding Algorithm

According to this algorithm (*18*), during each generation, the strings are paired randomly and crossed over. Assume that each string codes p parameters and that the strings A_i and A_j are crossed over to produce the strings A'_i and A'_j. If we denote with D_{mh} the mth parameter of the hth string, where $1 \leq m \leq p$, then the phenotypic distance d_{A_i,A_j} of the strings A_i and A_j is calculated according to the following:

$$d_{A_i,A_j} = \sqrt{\sum_{m=1}^{p} (D_{mj} - D_{mj})^2}$$

If $d_{A_i,A'_i} + d_{A_j,A'_j} < d_{A_i,A'_j} + d_{A_j,A'_i}$, then the string A_i is competing against the string A'_i, and the string A_j against the string A'_j; otherwise, the string A_i is competing against the string A'_j, and the string A_j against the string A'_i. The fittest of the competing strings are reproduced for the next generation.

Appendix IV. The Generic QC Rules

Let SD and m be the standard deviation and mean of the control measurements when the analytical process is in control, and let c, n, min, max, min_c, max_c, p, and a be parameters that are either user-defined or optimized. The generic rules of the program "Optimize" are:

1. The values/absolute values of c out of the last n control measurements are $\lesseqgtr m + (max)\ (SD)$ and (or) $\lesseqgtr m - (min)\ (SD)$.

2. The value/absolute value of the sum of the differences (control measurement $- m$) of the last n controls is $\lesseqgtr (max)\ (SD)$ and (or) $\lesseqgtr (min)\ (SD)$.

The sum can be an algebraic sum, a sum of absolute values, or a cumulative sum.

The calculation of the cumulative sum is initiated when a control measurement is $\lesseqgtr m + (max_c)\ (SD)$ and (or) $\lesseqgtr m - (min_c)\ (SD)$.

3. The range of the last n control measurements is $\lesseqgtr (max)\ (SD)$ and (or) $\lesseqgtr (min)\ (SD)$.

4. The value/absolute value of the difference between the mean of the last n control measurements and m is $\lesseqgtr (max)\ (SD)$ and (or) $\lesseqgtr (min)\ (SD)$.

5. The mean of the last n control measurements is different from m at the P level of significance.

6. The SD of the last n control measurements is $\lesseqgtr (max)\ (SD)$ and (or) $\lesseqgtr (min)\ (SD)$.

7. The SD of the last n control measurements is $> SD$ at the P level of significance.

8. The smoothed mean of the last n control measurements is different from m at the P level of significance, when the smoothing constant $= a$.

9. The smoothed SD of the last n control measurements is $> SD$ at the P level of significance, when the smoothing constant $= a$.

10. The value/absolute value of the tracking signal of the last n control measurements is $\lesseqgtr (max)\ (SD)$ and (or) $\lesseqgtr (min)\ (SD)$, when the smoothing constant $= a$.

The rules of the program "Design" are:

1. The absolute values of n out of the last n control measurements are $> m + (max)\ (SD)$.

Bounds of the parameters of rule 1: $1 \leq n \leq 4$, and $0 \leq max \leq 6.3$.

2. The range of the last n control measurements is $>(max)\ (SD)$.

3. The absolute value of the difference between the mean of the last n control measurements and m is $>(max)\ (SD)$.

4. The SD of the last n control measurements is $>(max)\ (SD)$.

Bounds of the parameters of rules 2, 3, and 4: $2 \leq n \leq 4$, and $0 \leq max \leq 6.3$. Since max is coded by a 6-bit substring, the accuracy of the calculation of max is $6.3/(2^6 - 1) = 0.1$.