A Study on Constrained MA Using GA and SQP: Analytical vs. Finite-Difference Gradients

S.D. Handoko, C.K. Kwoh, Y.S. Ong, and M.H. Lim

***Abstract*— Many deterministic algorithms in the context of constrained optimization require the first-order derivatives, or the gradient vectors, of the objective and constraint functions to determine the next feasible direction along which the search should progress. Although the second-order derivatives, or the Hessian matrices, are also required by some methods such as the *sequential quadratic programming* (SQP), their values can be approximated based on the first-order information, making the gradients central to the deterministic algorithms for solving constrained optimization problems. In this paper, two ways of obtaining the gradients are compared under the framework of the simple *memetic algorithm* (MA) employing *genetic algorithm* (GA) and SQP. Despite the simplicity and straightforwardness of the finite-difference gradients, faster convergence rate can be achieved when the analytical gradients can be made available. The savings on the number of function evaluations as well as the amount of time taken to solve some benchmark problems are presented along with some discussions.**

I. Introduction

Many optimization problems in practice are constrained: there ought to be some limitations that must not be violated. Generally, they can be formulated as finding a vector $\mathbf{x}$ of n real-valued independent variables that minimizes

$$f(\mathbf{x}) \tag{1}$$

subject to

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{0} \tag{2}$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{3}$$

$\mathbf{x} \in \Re^n$ is often referred to as the *solution*, $f : \Re^n \rightarrow \Re$ the *objective* function, and $\mathbf{g} : \Re^n \rightarrow \Re^{n_g}$ and $\mathbf{h} : \Re^n \rightarrow \Re^{n_h}$ the *inequality* and *equality constraint* functions, respectively. Additionally, there may exist bound constraints of the form $\mathbf{x}_\ell \leq \mathbf{x} \leq \mathbf{x}_u$ where $\mathbf{x}_\ell$ is the *lower* and $\mathbf{x}_u$ the *upper bound*. These can, however, be trivially converted into the following sets of linear inequality constraints, resembling (2).

$$\mathbf{x}_\ell - \mathbf{x} \leq \mathbf{0} \tag{4}$$

$$\mathbf{x} - \mathbf{x}_u \leq \mathbf{0} \tag{5}$$

The above formulation is an instance of the well-known *nonlinear programming* problem where some or all of the objective and/or constraint functions may not be linear. In the following two sections, three categories of the nonlinear programming solvers and two alternatives of obtaining the first-order derivatives are discussed. Some empirical results comparing both ways of calculating the gradients are then presented. A conclusion is finally drawn at the last section of this paper.

Handoko, S.D., Kwoh C.K., and Ong Y.S. are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (email: danielhandoko@pmail.ntu.edu.sg, asckkwoh@ntu.edu.sg, and, asysong@ntu.edu.sg, respectively). Lim M.H. is with the School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore 639798 (email: emhlim@ntu.edu.sg)

II. Nonlinear Programming Solvers

A. Sequential Quadratic Programming (SQP)

SQP, also known as the *successive* or *recursive quadratic programming*, represents the current state of the art in solving the general nonlinear programming problems [1]. It belongs to the class of deterministic algorithms called the *methods of feasible directions* [2]. Algorithms in this class proceed from one feasible solution to another in an iterative manner. At each iteration, a direction-finding problem is solved; the solution to which is then used to direct a line search. The algorithms of Zoutendijk [3][4][5] and the sequential linear programming approaches [6][7][8] are two examples of the other techniques adopting similar behavior. While these two procedures are prone to zigzagging and slow convergence, SQP enjoys a quadratic rate of convergence owing to the second-order functional approximations employed.

Using the Newton's method, SQP [9] solves directly the Karush-Kuhn-Tucker (KKT) optimality conditions [10][11] as shown in the following for a local optimum $\mathbf{x}^*$ of the nonlinear program.

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^{n_g} \mu_i \nabla g_i(\mathbf{x}^*) + \sum_{i=1}^{n_h} \nu_i \nabla h_i(\mathbf{x}^*) = \mathbf{0} \tag{6}$$

subject to

$$g_i(\mathbf{x}^*) \leq 0 \qquad i = 1, \ldots, n_g \tag{7}$$

$$h_i(\mathbf{x}^*) = 0 \qquad i = 1, \ldots, n_h \tag{8}$$

$$\mu_i g_i(\mathbf{x}^*) = 0 \qquad i = 1, \ldots, n_g \tag{9}$$

$$\mu_i \geq 0 \qquad i = 1, \ldots, n_g \tag{10}$$

At each iteration of the Newton's method, however, the accompanying subproblem turns out to be the minimization of a quadratic approximation to the Lagrangian function over a set of linear approximations to the constraint functions. Given an iterate $(\mathbf{x}^{(k)}, \mu^{(k)}, \nu^{(k)})$ where $\mu^{(k)} \geq \mathbf{0}$ and the unrestricted $\nu^{(k)}$ are, respectively, the Lagrange multiplier estimates for the inequality and the equality constraints, a quadratic program as shown below is solved at each major iteration of the SQP.

$$f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)})^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 L(\mathbf{x}^{(k)}) \mathbf{d} \tag{11}$$

subject to

$$g_i(\mathbf{x}^{(k)}) + \nabla g_i(\mathbf{x}^{(k)})^T \mathbf{d} \leq 0 \qquad i = 1, \ldots, n_g \tag{12}$$

$$h_i(\mathbf{x}^{(k)}) + \nabla h_i(\mathbf{x}^{(k)})^T \mathbf{d} = 0 \qquad i = 1, \ldots, n_h \tag{13}$$

where

$$\begin{aligned} \nabla^2 L(\mathbf{x}^{(k)}) = \nabla^2 f(\mathbf{x}^{(k)}) \\ + \sum_{i=1}^{n_g} \mu_i^{(k)} \nabla^2 g_i(\mathbf{x}^{(k)}) \\ + \sum_{i=1}^{n_h} \nu_i^{(k)} \nabla^2 h_i(\mathbf{x}^{(k)}) \end{aligned} \tag{14}$$

Solving the above quadratic program produces a direction vector $\mathbf{d}^{(k)}$ that is then used to form a new iterate $\mathbf{x}^{(k+1)}$ using the following line search formula.

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)} \tag{15}$$

To determine the appropriate step length α, an ℓ_1 penalty function can be used to penalize the objective values of the infeasible solutions [12][13].

Although SQP looks quite promising, its major drawback is clearly the requirement of the second-order derivatives $\nabla^2 L(\mathbf{x}^{(k)})$ to be computed at each major iteration. Not only the computational cost can be extremely expensive, but also the exact calculation of the Hessian of the Lagrangian may not produce a positive-definite matrix, causing the quadratic programming solver to break down. It is thus recommended to keep the Hessian matrix positive-definite even though it may be positive-indefinite at the solution point [12][13]. This can be achieved by updating the Hessian matrix using the following quasi-Newton approximation.

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \frac{\mathbf{q}^{(k)}[\mathbf{q}^{(k)}]^T}{[\mathbf{q}^{(k)}]^T \mathbf{p}^{(k)}} - \frac{\mathbf{H}^{(k)} \mathbf{p}^{(k)} [\mathbf{p}^{(k)}]^T \mathbf{H}^{(k)}}{[\mathbf{p}^{(k)}]^T \mathbf{H}^{(k)} \mathbf{p}^{(k)}} \tag{16}$$

where

$$\mathbf{p}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \tag{17}$$

$$\mathbf{q}^{(k)} = \nabla L(\mathbf{x}^{(k+1)}) - \nabla L(\mathbf{x}^{(k)}) \tag{18}$$

and

$$\nabla L(\mathbf{x}) = \nabla f(\mathbf{x}) + \sum_{i=1}^{n_g} \mu_i^{(k+1)} \nabla g_i(\mathbf{x}) + \sum_{i=1}^{n_h} \nu_i^{(k+1)} \nabla h_i(\mathbf{x}) \tag{19}$$

It can be shown that a superlinear convergence rate can still be achieved even after the above modification to the rudimentary SQP algorithm. Nevertheless, convergence to a local optimum is inherent to all deterministic optimization algorithms. In general, nonlinear programming problems in the context of global optimization are intractable [14]: it is impossible to design a deterministic algorithm that would outperform the exhaustive search in assuring the solution obtained to be the true global optimum.

B. Genetic Algorithm (GA)

GA [15], the most widely used technique in the class of randomized algorithms called the *evolutionary computing*, is a population-based meta-heuristic optimization procedure that was initially proposed with the intention to overcome the limitation of the deterministic optimization algorithms. Inspired by the natural inheritance of genetic materials and the natural selection in the course of biological evolution [16], crossover, mutation, and survival of the fittest are at the heart of all GAs. The simplest form of the algorithm assumes only one population evolved from one generation to the next following the steps described in Algorithm 1.

Algorithm 1 Simple GA

```
Initialize a population
Evaluate the fitness of each individual
while no stopping criteria have been fulfilled do
    Rank the individuals in accordance to
      their fitness values
    Select some individuals based on their rankings as
      the candidates for crossover and mutation
    Evolve the population through crossover, mutation,
      and elitism
    Evaluate the fitness of each individual
end while
```

Over the past decade, research works on the evolutionary computing in the context of constrained optimization include Homomorphous Mapping [17], Stochastic Ranking [18], the ASCHEA [19], Simple Multimembered Evolution Strategy (SMES) [20] that is known to have used the smallest number of fitness function evaluations (FFEs) so far, and some others (*e.g.* [21][22][23][24][25]). In [21], the ranking scheme used throughout our studies, as described by the following three points, was first proposed.

- The feasible solution is preferred to the infeasible one.
- Between two feasible solutions, the one having better objective value is preferred.
- Between two infeasible solutions, the one having less amount of constraint violations is preferred.

Where GA was not really the main focus of these works, the evolutionary operators proposed may still be applicable. A specifically-designed operator may accelerate the search to a certain extent. However, despite its ability to converge to the true global optimum, it is generally agreeable that GA may suffer from excessively slow convergence to locate the global optimum with sufficient precision due to its failure in exploiting local information [26].

C. Memetic Algorithm (MA)

Section II-A and II-B actually suggest that there are two central yet competing goals in the design of an optimization method. As discussed in [27], these are the *exploration* and the *exploitation*. While the exploration provides some reliable estimates of the global optimum by searching through the entire search space, the exploitation enhances each estimate

by focussing the search efforts on its local neighborhood in order to produce a sufficiently accurate global optimum. The exploration has been made possible by the incorporation of randomness, an intrinsic characteristic of the evolutionary algorithms, which is generally better than the brute force or the exhaustive search. The exploitation, on the other hand, is achievable thanks to the formal mathematical derivations that has led researchers to some deterministic procedures.

Inspired by the notion of memes [28], the cultural entities capable of refinements, MA balances both aspects through the hybridization of an evolutionary algorithm as the global search method and a deterministic optimization algorithm as the local search method. In its simplest form, MA interleaves the exploration with the exploitation one after the other as elaborated in Algorithm 2. This means that all individuals in the current population have to undergo some refinements before they are evolved to the next generation. As the result, not only MA is able to converge to the global optimum with sufficient precision, but also it searches more efficiently than its conventional counterparts [26].

Algorithm 2 Simple MA

```
Initialize a population
Evaluate the fitness of each individual
while no stopping criteria have been fulfilled do
  for each individual in the population do
    refine the individual through a local search
  end for
  Rank the individuals in accordance to
    their fitness values
  Select some individuals based on their rankings as
    the candidates for crossover and mutation
  Evolve the population through crossover, mutation,
    and elitism
  Evaluate the fitness of each individual
end while
```

III. First-Order Derivatives

In determining the feasible directions, most—if not all—deterministic algorithms utilize the first-order derivatives, or the gradient vectors, of the objective and constraint functions. SQP, in particular, also utilizes the second-order derivatives, or the Hessian matrices, of these functions. Even though the latter can be relaxed using the quasi-Newton approximation, the earlier must still be made available in order to achieve this relaxation.

The gradient of a function is the direction along which the greatest rate of increase of the value of the function occurs. It is mathematically written as

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial F(\mathbf{x})}{\partial x_1} \\ \frac{\partial F(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial F(\mathbf{x})}{\partial x_n} \end{bmatrix} \tag{20}$$

where $\frac{\partial F(\mathbf{x})}{\partial x_i} = \lim_{\lambda \to 0} \frac{F(x_1,\ldots,x_i+\lambda,\ldots,x_n)-F(x_1,\ldots,x_i,\ldots,x_n)}{\lambda}$ is the partial derivative of $F(\mathbf{x})$ with respect to the variable x_i, which is defined as the rate at which the value of $F(\mathbf{x})$ changes as a result of the infinitesimally small change λ in x_i. Even though there exist some methods that stochastically approximate the derivatives [29][30], we have focussed our studies only on the deterministic calculations of the gradients, keeping in mind that these produce reasonably small errors, if any.

A. Analytical Differentiation

The mathematical expression of the function is analyzed to derive the functional form of its derivatives, hence the term analytical differentiation. Using some derivative rules, symbolic differentiation is done either manually by hands or automatically using some computer programs. As the function becomes more complicated, the chain rule can be employed to simplify the calculations, which will normally result in the analytical gradient sharing some common sub-expressions.

To carry out analytical differentiation in the context of constrained optimization, one must obviously be equipped with the mathematical expressions of both the objective and constraint functions. However, once the analytical gradients are obtained, the computation can be done very quickly as has been shown for the case of the airfoil design problem [31]. In other fields, such as the molecular simulation in which it is desirable to find a molecular conformation with the lowest energy state, the gradient of the potential energy function is readily available in literatures [32].

B. Numerical Differentiation

In numerical differentiation, the infinitesimal change λ is substituted by an arbitrarily small step δ, called the finite difference. Using either one of the following formulae, the gradient is approximated to within a certain error tolerance expressed in term of the big-O notation below. The vector $\mathbf{e}_i$ is a unit vector pointing at the direction of x_i.

$$\frac{\partial F(\mathbf{x})}{\partial x_i} = \frac{F(\mathbf{x}+\delta\mathbf{e}_i) - F(\mathbf{x})}{\delta} + \mathcal{O}(\delta) \tag{21}$$

$$\frac{\partial F(\mathbf{x})}{\partial x_i} = \frac{-F(\mathbf{x}+2\delta\mathbf{e}_i) + 4F(\mathbf{x}+\delta\mathbf{e}_i) - 3F(\mathbf{x})}{2\delta} + \mathcal{O}(\delta^2) \tag{22}$$

It can be seen from the two equations above that all that is required to perform numerical differentiation is only the evaluation of the function $F(\mathbf{x})$. However, the latter equation obviously requires more function evaluations than the earlier does. Furthermore, it should be noted that the calculation needs to be repeated until the partial derivative with respect to all the n dimensions are obtained, posing the requirement of $n+1$ FFEs for the earlier equation and $2n+1$ FFEs for the latter. Throughout our study, the forward difference given by the earlier equation has been used whenever numerical differentiation is preferred.

IV. Empirical Studies

To assess the effectiveness of the analytical as opposed to the finite-difference gradients, we have conducted two sets of experiments employing GA and SQP under the simple MA framework over the benchmark problem **g01**, **g03**, **g04**, **g06**, **g08**, **g11**, and **g12**. These problems were solvable using the *Simple Multimembered Evolutionary Strategy* (SMES) [20] in 800 generations, or equivalently, 240000 fitness function evaluations (FFEs). The dimensionality of the problems and some theoretical expectancies over the necessary number of FFEs are presented in Table I. Should a problem be solvable within m FFEs when using the analytical gradients (AG), it would theoretically be solvable within $(n+1)m$ FFEs—where n is the dimensionality of the problem—when using the finite-difference gradients (FDG), enabling an amount of saving as much as

$$\frac{\#\text{FFEs}_{(\text{FDG})} - \#\text{FFEs}_{(\text{AG})}}{\#\text{FFEs}_{(\text{FDG})}} \times 100\% \tag{23}$$

TABLE I
Characteristics of the Benchmark Problems

Problem	Number of Dimensions	Expected #FFEs		Expected Saving
		AG	FDG	
g01	13	m	$14m$	92.86%
g03	10	m	$11m$	90.91%
g04	5	m	$6m$	83.33%
g06	2	m	$3m$	66.67%
g08	2	m	$3m$	66.67%
g11	2	m	$3m$	66.67%
g12	3	m	$4m$	75.00%

In Table II, some information regarding the cost required by the SMES to solve exactly the seven benchmark problems considered here is presented. The numbers of generations are as were reported in [20] and the equivalent numbers of FFEs are obtained from these numbers based on the fact that 300 FFEs were performed in each generation.

Throughout our experiments, a fixed population size of 50 individuals was used. Between the two sets of experiments, one set assumes that the analytical gradients (see Appendix) are readily available while the other assumes it is necessary to utilize the finite-difference gradients. Where the analytical gradients are readily available, a maximum of 100 FFEs per local search was allowed. For fairness, an additional budget of $100n$ FFEs—where n is the number of dimensions—was provided whenever the finite-difference gradients were the choice, allowing up to $100(n+1)$ FFEs per local search.

TABLE II
Number of Generations and the Equivalent Number of FFEs Required by the SMES to Reach the Global Optimum

Problem	#Generations		#FFEs	
	Minimum	Average	Minimum	Average
g01	634	671	190200	201300
g03	41	184	12300	55200
g04	113	129	33900	38700
g06	47	249	14100	74700
g08	11	18	3300	5400
g11	28	88	8400	26400
g12	63	77	18900	23100

V. Results and Discussions

To locate the global optimum of each benchmark problem, the minimum and average numbers of FFEs required, over 30 independent runs, by the SMES and the simple MA using the analytical gradients (MA-AG) as well as the finite-difference gradients (MA-FDG) are tabulated in Table III and Table IV, respectively. The savings, expressed in term of percentages, on the average numbers of FFEs with respect to the SMES are then summarized in Table V.

TABLE III
The Minimum Number of FFEs over 30 Independent Runs Required to Reach the Global Optimum

Problem	SMES	MA-AG	MA-FDG
g01	190200	5186	71331
g03	12300	5155	55760
g04	33900	5154	30429
g06	14100	5191	15377
g08	3300	108	222
g11	8400	3940	7243
g12	18900	2763	10905

From these tables, it is clear that the MA-AG was always superior while the MA-FDG was sometimes inferior—both in the best and average cases—to the SMES. Deterioration of up to 195.24% was observed on the average performance of the MA-FDG with respect to the SMES, making the simple MA not always the best alternative to the pure evolutionary algorithms, particularly when the first-order derivatives need to be calculated numerically using the finite differences.

TABLE IV
THE AVERAGE NUMBER OF FFES OVER 30 INDEPENDENT RUNS REQUIRED TO REACH THE GLOBAL OPTIMUM

Problem	SMES	MA-AG	MA-FDG
g01	201300	9520	131763
g03	55200	6379	64360
g04	38700	5159	30460
g06	74700	8728	26477
g08	5400	2862	8365
g11	26400	5089	14790
g12	23100	17505	68200

TABLE V
THE PERCENTAGE OF SAVING ON THE AVERAGE NUMBER OF FFES WITH RESPECT TO THE SMES

Problem	MA-AG	MA-FDG
g01	95.27%	34.54%
g03	88.44%	−16.59%
g04	86.67%	21.29%
g06	88.32%	64.56%
g08	47.00%	−54.91%
g11	80.72%	43.98%
g12	24.22%	−195.24%

TABLE VI
THE AVERAGE CPU TIME (IN SECONDS) OVER 30 INDEPENDENT RUNS REQUIRED TO REACH THE GLOBAL OPTIMUM

Problem	MA-FDG	MA-AG
g01	19.329	1.227
g03	7.275	0.855
g04	3.734	0.977
g06	2.767	1.212
g08	1.229	0.556
g11	1.354	0.575
g12	8.592	2.259

TABLE VII
THE PERCENTAGE OF SAVING ACHIEVABLE BY USING THE MA-AG WITH RESPECT TO USING THE MA-FDG

Problem	Expected	#FFEs	CPU Time
g01	92.86%	92.77%	93.65%
g03	90.91%	90.09%	88.25%
g04	83.33%	83.06%	73.84%
g06	66.67%	67.04%	56.20%
g08	66.67%	65.79%	54.76%
g11	66.67%	65.59%	57.53%
g12	75.00%	74.33%	73.71%

As function evaluations are expected to take longer time when the gradients are also computed, we have considered for completeness the actual CPU time taken when carrying out the experiments on a 2.4 GHz quad-core processor with 8 GB RAM. Table VI tabulates the average CPU time taken over the 30 independent runs to perform the necessary FFEs to solve the problems. Table VII then summarizes the amount of savings achievable by the MA-AG in comparison with the MA-FDG. The percentage of savings on the average number of FFEs and the average CPU time are presented along with the expected savings from Section IV.

Clearly, it is deducible that the presence of the analytical gradients has provided significant cost reduction up to more than 90%. The savings on the number of FFEs were observed to match closely our expectations. Nevertheless, the savings on the CPU time are somewhat lower than expected due to the additional cost incurred when computing the analytical gradients. However, this was not true for the problem **g01**, the entirely linear constraint functions of which may have contributed largely to the comparable amount of savings on the number of FFEs and the amount of CPU time required to solve the problem. Furthermore, another observable trend is that the savings become more apparent as the number of dimensions increases as depicted in Figure 1 and 2. This is in line with the theoretical expectancies built in Section IV, which are represented by the dotted curves in both figures. The problem **g01**, **g03**, **g04**, **g06**, **g08**, **g11**, and **g12** are, respectively, denoted by point (•), plus sign (+), cross (×), circle (○), square (□), diamond (◊), and star (⋆) in these figures.

In many practical problems, for which a sufficiently large number of dimensions is common and a single FFE can be extremely expensive [33], it is very likely that the MA-FDG will lose out from the MA-AG. While a single evaluation of the gradients using the finite differences takes n times longer than a single FFE would take—where n is the dimensionality of the problem—computing the analytical gradients will not normally incur as much time requirement. It has been shown that the airfoil design problem, for example, requires up to only twice the time complexity of its single FFE [31].

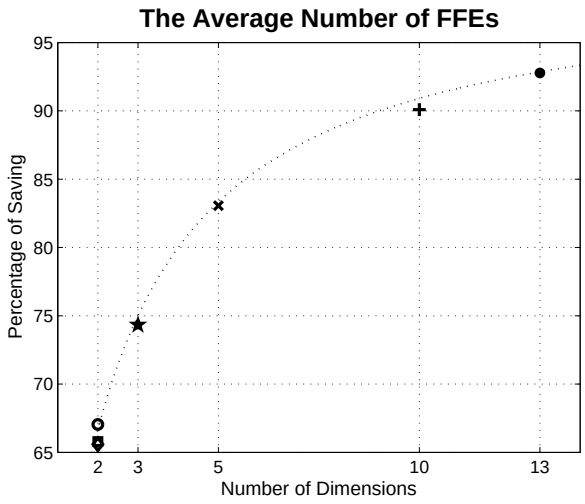

Fig. 1. The percentage of saving on the average number of FFEs achievable by using the MA-AG with respect to using the MA-FDG

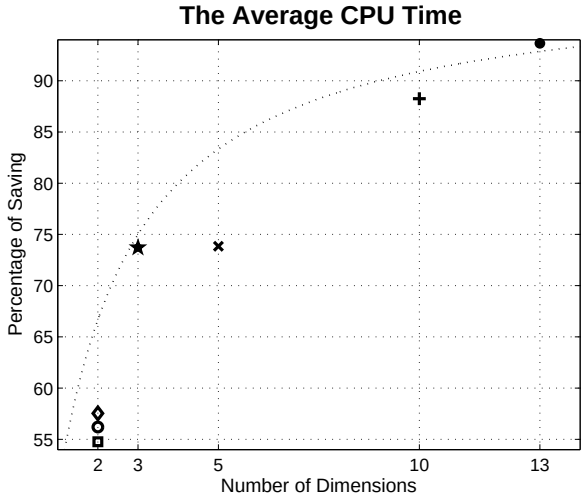

Fig. 2. The percentage of saving on the average CPU time achievable by using the MA-AG with respect to using the MA-FDG

VI. Conclusion

From our comparative studies, we can conclude that the analytical gradients, whenever available at a reasonable cost, shall be utilized in the course of searching for the global optimum for which some deterministic algorithms in the context of constrained optimization are involved. We have shown that using finite-difference gradients bears the risk of deteriorating the performance of the simple MA over the pure evolutionary algorithm. However, the risk is inexistent when the analytical gradients are used.

Appendix

g01 [34]:

$$f(\mathbf{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \begin{cases} 5 - 10x_i & i = 1,2,3,4 \\ -1 & \text{otherwise} \end{cases}$$

$$g_1(\mathbf{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10$$

$$\frac{\partial g_1(\mathbf{x})}{\partial x_i} = \begin{cases} 2 & i = 1,2 \\ 1 & i = 10,11 \\ 0 & \text{otherwise} \end{cases}$$

$$g_2(\mathbf{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10$$

$$\frac{\partial g_2(\mathbf{x})}{\partial x_i} = \begin{cases} 2 & i = 1,3 \\ 1 & i = 10,12 \\ 0 & \text{otherwise} \end{cases}$$

$$g_3(\mathbf{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10$$

$$\frac{\partial g_3(\mathbf{x})}{\partial x_i} = \begin{cases} 2 & i = 2,3 \\ 1 & i = 11,12 \\ 0 & \text{otherwise} \end{cases}$$

$$g_4(\mathbf{x}) = -8x_1 + x_{10}$$

$$\frac{\partial g_4(\mathbf{x})}{\partial x_i} = \begin{cases} -8 & i = 1 \\ 1 & i = 10 \\ 0 & \text{otherwise} \end{cases}$$

$$g_5(\mathbf{x}) = -8x_2 + x_{11}$$

$$\frac{\partial g_5(\mathbf{x})}{\partial x_i} = \begin{cases} -8 & i = 2 \\ 1 & i = 11 \\ 0 & \text{otherwise} \end{cases}$$

$$g_6(\mathbf{x}) = -8x_3 + x_{12}$$

$$\frac{\partial g_6(\mathbf{x})}{\partial x_i} = \begin{cases} -8 & i = 3 \\ 1 & i = 12 \\ 0 & \text{otherwise} \end{cases}$$

$$g_7(\mathbf{x}) = -2x_4 - x_5 + x_{10}$$

$$\frac{\partial g_7(\mathbf{x})}{\partial x_i} = \begin{cases} -2 & i = 4 \\ -1 & i = 5 \\ 1 & i = 10 \\ 0 & \text{otherwise} \end{cases}$$

$$g_8(\mathbf{x}) = -2x_6 - x_7 + x_{11}$$

$$\frac{\partial g_8(\mathbf{x})}{\partial x_i} = \begin{cases} -2 & i = 6 \\ -1 & i = 7 \\ 1 & i = 11 \\ 0 & \text{otherwise} \end{cases}$$

$$g_9(\mathbf{x}) = -2x_8 - x_9 + x_{12}$$

$$\frac{\partial g_9(\mathbf{x})}{\partial x_i} = \begin{cases} -2 & i = 8 \\ -1 & i = 9 \\ 1 & i = 12 \\ 0 & \text{otherwise} \end{cases}$$

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

g03 [35]:

$$f(\mathbf{x}) = -\left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i$$
$$\frac{\partial f(\mathbf{x})}{\partial x_i} = -\left(\sqrt{n}\right)^n \prod_{\substack{j=1 \\ j\neq i}}^{n} x_i$$
$$h(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 - 1$$
$$\frac{\partial h(\mathbf{x})}{\partial x_i} = 2x_i$$

where $n = 10$ and $0 \leq x_i \leq 1$ for $i = 1, 2, \ldots, n$.

g04 [36]:

$$f(\mathbf{x}) = 0.8356891x_1x_5 + 5.3578547x_3^2 + 37.293239x_1 - 40792.141$$
$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \begin{cases} 0.8356891x_5 + 37.293239 & i = 1 \\ 10.7157094x_3 & i = 3 \\ 0.8356891x_1 & i = 5 \\ 0 & \text{otherwise} \end{cases}$$
$$g_1(\mathbf{x}) = 0.0006262x_1x_4 + 0.0056858x_2x_5 - 0.0022053x_3x_5 - 6.665593$$
$$\frac{\partial g_1(\mathbf{x})}{\partial x_i} = \begin{cases} 0.0006262x_4 & i = 1 \\ 0.0056858x_5 & i = 2 \\ -0.0022053x_5 & i = 3 \\ 0.0006262x_1 & i = 4 \\ 0.0056858x_2 - 0.0022053x_3 & i = 5 \end{cases}$$
$$g_2(\mathbf{x}) = -0.0006262x_1x_4 - 0.0056858x_2x_5 + 0.0022053x_3x_5 - 85.334407$$
$$\frac{\partial g_2(\mathbf{x})}{\partial x_i} = \begin{cases} -0.0006262x_4 & i = 1 \\ -0.0056858x_5 & i = 2 \\ 0.0022053x_5 & i = 3 \\ -0.0006262x_1 & i = 4 \\ -0.0056858x_2 + 0.0022053x_3 & i = 5 \end{cases}$$
$$g_3(\mathbf{x}) = 0.0029955x_1x_2 + 0.0071317x_2x_5 + 0.0021813x_3^2 - 29.48751$$
$$\frac{\partial g_3(\mathbf{x})}{\partial x_i} = \begin{cases} 0.0029955x_2 & i = 1 \\ 0.0029955x_1 + 0.0071317x_5 & i = 2 \\ 0.0043626x_3 & i = 3 \\ 0 & i = 4 \\ 0.0071317x_2 & i = 5 \end{cases}$$
$$g_4(\mathbf{x}) = -0.0029955x_1x_2 - 0.0071317x_2x_5 - 0.0021813x_3^2 + 9.48751$$
$$\frac{\partial g_4(\mathbf{x})}{\partial x_i} = \begin{cases} -0.0029955x_2 & i = 1 \\ -0.0029955x_1 - 0.0071317x_5 & i = 2 \\ -0.0043626x_3 & i = 3 \\ 0 & i = 4 \\ -0.0071317x_2 & i = 5 \end{cases}$$
$$g_5(\mathbf{x}) = 0.0012547x_1x_3 + 0.0019085x_3x_4 + 0.0047026x_3x_5 - 15.699039$$
$$\frac{\partial g_5(\mathbf{x})}{\partial x_i} = \begin{cases} 0.0012547x_3 & i = 1 \\ 0 & i = 2 \\ 0.0012547x_1 + 0.0019085x_4 & \\ \quad +0.0047026x_5 & i = 3 \\ 0.0019085x_3 & i = 4 \\ 0.0047026x_3 & i = 5 \end{cases}$$
$$g_6(\mathbf{x}) = -0.0012547x_1x_3 - 0.0019085x_3x_4 - 0.0047026x_3x_5 + 10.699039$$
$$\frac{\partial g_6(\mathbf{x})}{\partial x_i} = \begin{cases} -0.0012547x_3 & i = 1 \\ 0 & i = 2 \\ -0.0012547x_1 - 0.0019085x_4 & \\ \quad -0.0047026x_5 & i = 3 \\ -0.0019085x_3 & i = 4 \\ -0.0047026x_3 & i = 5 \end{cases}$$

where $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, and $27 \leq x_i \leq 45$ for $i = 3, 4, 5$.

g06 [34]:

$$f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$
$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \begin{cases} 3(x_i - 10)^2 & i = 1 \\ 3(x_i - 20)^2 & i = 2 \end{cases}$$
$$g_1(\mathbf{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100$$
$$\frac{\partial g_1(\mathbf{x})}{\partial x_i} = -2x_i + 10$$
$$g_2(\mathbf{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81$$
$$\frac{\partial g_2(\mathbf{x})}{\partial x_i} = \begin{cases} 2x_i - 12 & i = 1 \\ 2x_i - 10 & i = 2 \end{cases}$$

where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$.

g08 [17]:

$$f(\mathbf{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$
$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \begin{cases} -\frac{6\pi \sin^2(2\pi x_1)\cos(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)} & \\ \quad +\frac{3\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^4(x_1+x_2)} & \\ \quad +\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)^2} & i = 1 \\ -\frac{2\pi \sin^3(2\pi x_1)\cos(2\pi x_2)}{x_1^3(x_1+x_2)} & \\ \quad +\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)^2} & i = 2 \end{cases}$$

$$g_1(\mathbf{x}) = x_1^2 - x_2 + 1$$
$$\frac{\partial g_1(\mathbf{x})}{\partial x_i} = \begin{cases} 2x_1 & i = 1 \\ -1 & i = 2 \end{cases}$$
$$g_2(\mathbf{x}) = 1 - x_1 + (x_2 - 4)^2$$
$$\frac{\partial g_2(\mathbf{x})}{\partial x_i} = \begin{cases} -1 & i = 1 \\ 2x_2 - 8 & i = 2 \end{cases}$$

where $0 \leq x_i \leq 10$ for $i = 1, 2$.

g11 [17]:

$$f(\mathbf{x}) = x_1^2 + (x_2 - 1)^2$$
$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \begin{cases} 2x_1 & i = 1 \\ 2x_2 - 2 & i = 2 \end{cases}$$
$$h(\mathbf{x}) = -x_1^2 + x_2$$
$$\frac{\partial h(\mathbf{x})}{\partial x_i} = \begin{cases} -2x_1 & i = 1 \\ 1 & i = 2 \end{cases}$$

where $-1 \leq x_i \leq 1$ for $i = 1, 2$.

g12 [17]:

$$f(\mathbf{x}) = \frac{(x_1 - 5)^2 + (x_2 - 5)^2 + (x_3 - 5)^2 - 100}{100}$$
$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{2x_i - 10}{100}$$
$$g(\mathbf{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625$$
$$\frac{\partial g(\mathbf{x})}{\partial x_i} = \begin{cases} 2(x_1 - p) & i = 1 \\ 2(x_2 - q) & i = 2 \\ 2(x_3 - r) & i = 3 \end{cases}$$

where $0 \leq x_i \leq 10$ for $i = 1, 2, 3$ and $p, q, r = 1, 2, \ldots, 9$. A solution $\mathbf{x}$ is feasible if and only if there exist p, q, r such that the above inequality holds.

REFERENCES

[1] Schittkowski, K., "NLPQL: A FORTRAN Subroutine Solving Constrained Nonlinear Programming Problems," *Annals of Operations Research*, 5(2), pp. 485–500, 1986.

[2] Bazaraa, M.S., H.D. Sherali, and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, Wiley-Interscience, 2006.

[3] Zoutendijk, G., *Methods of Feasible Directions*, Elsevier, 1960.

[4] Topkis, D.M. and A.F. Veinott, "On the Convergence of Some Feasible Diretion Algorithms for Nonlinear Programming," *SIAM Journal on Control*, 5(2), pp. 268–279, 1967.

[5] Zangwill, W.I., *Nonlinear Programming: A Unified Approach*, Prentice-Hall, 1969.

[6] Griffith, R.E. and R.A. Stewart, "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," *Management Science*, 7(4), pp. 379–392, 1961.

[7] Baker, T.E. and L.S. Lasdon, "Successive Linear Programming at Exxon," *Management Science*, 31(3), pp. 264–274, 1985.

[8] Zhang, J.Z., N.H. Kim, and L.S. Lasdon, "An Improved Successive Linear Programming Algorithm," *Management Science*, 31(10), pp. 1312–1331, 1985.

[9] Wilson, R.B., "A Simplicial Algorithm for Convex Programming," Ph.D. Thesis, Harvard University, 1963.

[10] Karush, W., "Minima of Functions of Several Variables with Inequalities as Side Conditions," M.Sc. Dissertation, Department of Mathematics, University of Chicago, 1939.

[11] Kuhn, H.W. and A.W. Tucker, "Nonlinear Programming," in *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pp. 481–492, 1950.

[12] Han, S.P., "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems," *Mathematical Programming*, 11(1), pp. 263–282, 1976.

[13] Powell, M.J.D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," in *Lecture Notes in Mathematics*, 630, Springer-Verlag, 1978.

[14] Wright, S.J., "Nonlinear and Semidefinite Programming," in *Proceedings of Symposia in Applied Mathematics*, 61, pp. 115–138, 2004.

[15] Holland, J.H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[16] Darwin, C., *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, John Murray, 1859.

[17] Koziel, S. and Z. Michalewicz, "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization," *Evolutionary Computation*, 7(1), pp. 19–44, 1999.

[18] Runarsson, T.P. and Y. Xin, "Stochastic Ranking for Constrained Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation*, 4(3), pp. 284–294, 2000.

[19] Hamida, S.B. and M. Schoenauer, "ASCHEA: New Results Using Adaptive Segregational Constraint Handling," in *Proceedings of the 2002 Congress on Evolutionary Computation*, pp. 884–889, 2002.

[20] Mezura-Montes, E. and C.A.C. Coello, "A Simple Multi-Membered Evolution Strategy to Solve Constrained Optimization Problems," *IEEE Transactions on Evolutionary Computation*, 9(1), pp. 1–17, 2005.

[21] Deb, K., "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering*, 186(2–4), pp. 311–338, 2000.

[22] Barbosa, H.J.C. and A.C.C. Lemonge, "A New Adaptive Penalty Scheme for Genetic Algorithms," *Information Science*, 156(3–4), pp. 215–251, 2003.

[23] Farmani, R. and J.A. Wright, "Self-Adaptive Fitness Formulation for Constrained Optimization," *IEEE Transactions on Evolutionary Computation*, 7(5), pp. 445–455, 2003.

[24] Chootinan, P. and A. Chen, "Constraint Handling in Genetic Algorithms Using a Gradient-based Repair Method," *Computers and Operation Research*, 33(8), pp. 2263–2281, 2006.

[25] Elfeky, E.Z., R.A. Sarker, and D.L. Essam, "A Simple Ranking and Selection for Constrained Evolutionary Optimization," in *Lecture Notes in Computer Science*, 4247, Springer-Verlag, 2006.

[26] Tang, J., M.H. Lim, and Y.S. Ong, "Diversity-Adaptive Parallel Memetic Algorithm for Solving Large Scale Combinatorial Optimization Problems," *Soft Computing: A Fusion of Foundations, Methodologies, and Applications*, 11(9), pp. 873–888, 2007.

[27] Torn, A. and A. Zilinskas, *Global Optimization*, Springer-Verlag, 1989.

[28] Dawkins, R., *The Selfish Gene*, Oxford University Press, 1976.

[29] Spall, J.C., "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Transactions on Automatic Control*, 37(3), pp. 332–341, 1992.

[30] Spall, J.C., "Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization," *IEEE Transactions on Aerospace and Electronic Systems*, 34(3), pp. 817–823, 1998.

[31] Ong, Y.S., K.Y. Lum, and P.B. Nair, "Evolutionary Algorithm with Hermite Radial Basis Function Interpolants for Computationally Expensive Adjoint Solvers," *Computational Optimization and Applications*, Online First, 2007.

[32] Levitt, M., "Conformation Analysis of Proteins," Ph.D. Thesis, University of Cambridge, 1971.

[33] Ong, Y.S., P.B. Nair, and A.J. Keane, "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling," *American Institute of Aeronautics and Astronautics Journal*, 41(4), pp. 687–696, 2003.

[34] Floudas, C.A. and P.M. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer-Verlag, 1990.

[35] Michalewicz, Z., G. Nazhiyath, and M. Michalewicz, "A Note on Usefulness of Geometrical Crossover for Numerical Optimization Problems," in *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pp. 305–312, 1996.

[36] Himmelblau, D.M., *Applied Nonlinear Programming*, McGraw-Hill, 1972.