Comput Econ (2016) 48:473–485
DOI 10.1007/s10614-015-9522-7

On the Choice of a Genetic Algorithm for Estimating GARCH Models

Manuel Rizzo[1] · **Francesco Battaglia**[1]

Accepted: 18 August 2015 / Published online: 25 August 2015

Abstract The GARCH models have been found difficult to build by classical methods, and several other approaches have been proposed in literature, including metaheuristic and evolutionary ones. In the present paper we employ genetic algorithms to estimate the parameters of GARCH(1,1) models, assuming a fixed computational time (measured in number of fitness function evaluations) that is variously allocated in number of generations, number of algorithm restarts and number of chromosomes in the population, in order to gain some indications about the impact of each of these factors on the estimates. Results from this simulation study show that if the main purpose is to reach a high quality solution with no time restrictions the algorithm should not be restarted and an average population size is recommended, while if the interest is focused on driving rapidly to a satisfactory solution then for moderate population sizes it is convenient to restart the algorithm, even if this means to have a small number of generations.

Keywords Evolutionary computation · Conditional heteroscedasticity · Parameter estimation · Restarts

1 Introduction

The class of generalized autoregressive conditional heteroscedastic (GARCH) models, introduced by Bollerslev (1986), has received great attention in the literature devoted

✉ Manuel Rizzo
manuel.rizzo@uniroma1.it

Francesco Battaglia
francesco.battaglia@uniroma1.it

[1] Department of Statistics, Sapienza University of Rome, Piazzale Aldo Moro 5, 00100 Roma, Italy

to the analysis of financial time series, because of its importance in reproducing the so-called *volatility clustering*, along with ARCH models (Engle 1982), of which the GARCH models represent, indeed, a generalization.

Building models (precisely estimating their parameters) with this features is not trivial: in fact, since the earliest related works, the inference from GARCH models has always been based on the Maximum Likelihood Estimation principle, that, under some Gaussian assumptions, leads to the Quasi Maximum Likelihood Estimator (QMLE), which is the most used in this field. The QML function, even for the simplest form of the model, the GARCH(1,1), has been found numerically difficult to optimize using classical methods, like Newton's for example, because of drawbacks like multimodality (see Zumbach 2000 for an account of these problems), so different kind of optimization methods have been proposed.

The growing importance in literature of metaheuristics and evolutionary procedures has encouraged many researchers to try using these methods in statistical applications (for some comprehensive accounts see Baragona et al. 2011 and Winker and Gilli 2004). Several metaheuristics and evolutionary algorithms have already been proposed in literature for the estimating problem in exam. As observed by Winker and Maringer (2009) the theoretical ML estimator for the GARCH model cannot be observed in practice. Deterministic algorithms approximations very often provide high quality solutions, but they fail to do so from time to time due to the inherent complexity of the estimation problem. This is true even for the simplest form of the model, the GARCH(1,1). For such a reason stochastic algorithms may be more efficient. Winker and Maringer proposed the Threshold Accepting, but many more options can be found in literature. Adanu (2006) showed that the performance of the genetic algorithm (GA) and the Differential Evolution in optimizing the GARCH model, compared with two local search methods, is competitive, especially when the problem complexity is high. Wang and Li (2001) observed that the GA based GARCH optimization outperforms the conventional numerical methods on the aspect of computational robustness and accuracy. Furthermore, the GAs have been successfully employed in several applications where modifications of the basic GARCH model are involved, for example Fuzzy-GARCH models (Hung 2009) or Grey GARCH-Type models (Geng and Zhang 2015). Santamaría-Bonfil et al. (2015), proposed a Support Vector Machine model and used a hybrid genetic algorithm for estimating the parameters.

In this paper we examine the behaviour of GAs when used to estimate the parameters of a GARCH(1,1) when a fixed computational time, measured in number of fitness function evaluations, is allocated in number of generations G, number of algorithm restarts R and number of chromosomes in the population N, in a range of different ways. Thus we can study and analyze the effects of each of these parameters on the estimates.

2 GARCH(1,1) Estimation and GAs

The GARCH(1,1) model is defined by the following equations:

$$\begin{aligned} Y_t &= \sqrt{h_t}\epsilon_t, \\ h_t &= \psi_1 + \psi_2 Y_{t-1}^2 + \psi_3 h_{t-1}, \end{aligned}$$

with the conditions:

$$\psi_1 > 0,\ \ \psi_2, \psi_3 \geq 0,\ \ \psi_2 + \psi_3 < 1,$$

where Y_t is a zero mean stochastic process, ϵ_t a stochastic process composed by independent standard normal random variables, h_t the conditional variance of Y_t and $\underline{\psi} = (\psi_1, \psi_2, \psi_3)$ the model parameters vector.

The presence of the conditional variance h_t allows the model to account for the volatility of the process, but the estimation of $\underline{\psi}$ is not easy. The standard techniques maximize the QML function:

$$L(\underline{\psi}; \underline{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{n}\left[\log(h_t) + \frac{y_t^2}{h_t}\right], \tag{1}$$

where $\underline{y} = (y_1, ..., y_n)$ is the observed time series.

The difficulties experienced in maximizing this function with classical methods have induced researchers to study other optimization tools, and GAs represent a valid option. In particular, GAs allow to easily select the quality of the approximation of the solution, and this is very important when dealing with a large number of financial datasets.

The GA was introduced by Holland (1975) and became, over the years, one of the most important algorithm in the metaheuristic and evolutionary field, because of its simplicity and variety of applications. Many modifications of the GA have been proposed in literature, but the basic GA can be summarized as follows: the solutions of the problem are represented as individuals, called *chromosomes*, which are binary coded vectors composed by values called *genes*, of a population that evolves through generations to populations of better individuals; an individual is considered better than another if he is better able to adapt to the environment, and this is measured by the fitness function, to be optimized. The process of evolution in every generation takes place by applying, on individuals selected stochastically from the population according to their fitness, two genetic operators: *crossover,* that allows with a fixed probability two individuals to reproduce and generate two new solutions (children), which are created by randomly choosing a common cutting point from the original chromosomes and taking the left part from the first parent and the right part from the other and vice versa, and *mutation*, whereby every gene of chromosomes in the population is subject to change its value, simulating the random mutations in nature, with a fixed probability. The flow of generations generally stops if it has reached a prior fixed number, if there are no more significative improvements within a certain number of generations or if a stopping criterion is met.

The algorithm is initialized by generating a random initial population, and this may influence qualitatively and quantitatively the convergence to the optimum: in fact the lack of a mechanism that can dynamically guarantee a certain degree of diversity in the population may lead the algorithm to prematurely converge to local optima. Several methods have been proposed in literature to deal with this problem, for example the *restart* mechanism.

The *restart* mechanism is generally employed when an optimization algorithm has not found a significative improvement of the current solution within a fixed number of iterations or if it has reached a local optimum: in this case the algorithm is restarted with a new initial solution, with the purpose of improving the speed of convergence to the global optimum and escaping the local ones.

Ghannadian et al. (1996) brought formally this idea in the field of GAs, proposing the restart mechanism as a replacement for the mutation operator: as the search stagnates, a new initial population of chromosomes is created and the algorithm is restarted. They studied analytically the expected time for the algorithm to reach an optimal solution using the Markov Chain theory. Fukunaga (1998) proposed a distinction between *dynamic restart*, the one just described (without the elimination of the mutation operator), and *static restart*, where the time at which the restart is applied is prior fixed: in the classic Traveling Salesman Problem, he used a static restart strategy based on performance data from previous runs of the GA on similar problems, when a fixed computational time is given. The results showed the competitivity of the static strategy in relation to a dynamic one. It is worth noting that the basic dynamic restart can be thought as a particular case of the *random immigrants scheme* (introduced by Grefenstette 1992), another method that allows to deal with the premature convergence problem, for which the solutions regeneration step takes place only if a certain condition is met and it may be applied to the population as a whole. For other different applications of dynamic restart, that avoid premature convergence and improve the fitness, see Misevicius (2009) and Phanden et al. (2012). The restart method used in the present paper is similar to the version analyzed by Winker (2006) in the field of the Threshold Accepting, and it may be classified as static, because an equal amount of time is allocated to every restart; in such a way we can find some indications about the optimal number of restart.

As far as the choice of the population size is concerned, both empirical and theoretical studies have been conducted (see, respectively, Alander 1992 and Goldberg (1989) for examples), but a general indication tells that it should be large enough to allow an exhaustive search, but not too large to avoid heavy computational efforts. As far as the usual population sizes are of the order of the tens, in this paper a range of population sizes from 10 to 100 is considered.

Lastly, the importance of the number of generations has been shown by Rudolph (1997) in his convergence theorem valid for *elitist* GAs: it states that if we choose a non-zero mutation rate and we employ the *elitism,* a strategy based on saving the best result of every generation to obtain a monotonic *best fitness in generation* function, then the GA will converge almost surely to the global optimum when the number of generations tends to infinity. For this reason we shall employ the elitist strategy in the present analysis.

3 Problem Description

The problem studied in the present paper is based on a paper by Winker (2006), where he provides a formal theoretical framework for the analysis of the Threshold Accepting method, and then analyzes the optimal allocation of an amount of computational resources C (measured in iterations) on number of iterations I and number of restarts R of the algorithm, in an application to the uniform design problem. He shows that the optimal choice is neither a large number of restarts (with few iterations) nor a very small one (with many iterations): rather the optimum seems to be of the order of 10–20 restarts.

In a GA analysis we must consider also the population size as a factor influencing time, because it is a *population based* method (in fact in every iteration a population of solutions is considered). Thus, the question becomes how to distribute a fixed computational time C in number of restarts R, number of generations G and number of chromosomes in the population N, with the constraint $C = R \times G \times N$, meaning that this computational time will be measured in number of fitness function evaluations (which is usually the most computationally expensive step).

Now we shall describe first the implementation of a GA for a GARCH(1,1) parameters estimation, then the choices and the objectives for the time allocation problem will be discussed.

3.1 GA's Implementation

3.1.1 Coding

Every chromosome in the population will be composed by 21 genes and will represent an estimate of the vector $\underline{\psi}$ by use of the binary coding: genes 1–7 will indicate $\hat{\psi}_1$, genes 8–14 $\hat{\psi}_2$, genes 15–21 $\hat{\psi}_3$. We shall adopt the usual rule to represent a parameter θ defined on the real interval $[a, b]$ by binary coding:

$$\theta = a + \frac{b-a}{2^M - 1} \sum_{j=1}^{M} 2^{j-1} x_j,$$

where M is the number of genes in the chromosome and x_j is the value of the j-th gene.

ψ_1 will be defined on $[0, V]$, where V is the unconditional variance of the observed time series; ψ_2 on $[0, 1]$; ψ_3 on $[0, (1 - \psi_2)]$, to include the constraint $\psi_2 + \psi_3 < 1$. If a decoded chromosome provides an unacceptable value (the rare eventualities are $\psi_1 = 0$ or $\psi_2 + \psi_3 = 1$) it is rejected and regenerated.

3.1.2 Genetic Operators

The selection mechanism employed is the *roulette wheel*, so that every individual has a probability of selection for the evolution proportional to his fitness; we also employ the *single-point crossover* as reproduction operator, with random cutting point and rate 0.7, and the *bit-flip mutation*, with rate 0.1, chosen on the basis of some pilot experiments. Lastly, the elitist strategy is employed, so the best individual in the previous generation will replace the worst in the current, if there was no improvement of the best fitness in the current generation.

3.1.3 Fitness

The fitness function f to be maximized will be based on the log-likelihood (1), divided by a constant $\beta = 800$ to avoid troubles related to numerical approximations and using the exponential scale to avoid negative values that give problems to the selection mechanism:

$$f(\underline{\psi}) = \exp\{L(\underline{\psi}; \underline{y})/\beta\}. \tag{2}$$

3.1.4 Restart Mechanism

The number of restarts R for every time allocation is prior fixed and the restart mechanism will operate sequentially, so that at the end of the runs only a vector of fitness values will result: precisely, at the end of the GA run R vectors of best fitness in generation values will result but, for every generation, only the best value between all the restarts will be selected, while the others will be lost. In this way we operate as if we ran several parallel GAs keeping only the best value between the restarts for every generation.

3.1.5 Data

The time series analyzed are simulated according to a GARCH(1,1) model with $\underline{\psi}$ held fixed, which is considered the true parameters vector of the model, and the GA will operate to reach the global optimum in $f(\hat{\underline{\psi}}_{MLE})$, that is the QMLE estimator maximizing (2). Every time series $\underline{y}$ leads to a different value of $f(\hat{\underline{\psi}}_{MLE})$, so we will proceed in the following way: for every time allocation 50 simulated time series of length $n = 500$ are considered and the GA will operate on each of them; every $f(\hat{\underline{\psi}}^{(g)}_{GA}(\underline{y}))$ (namely the best fitness in generation g obtained with the series $\underline{y}$) will be divided by $f(\hat{\underline{\psi}}_{MLE}(\underline{y}))$, and at the end of the runs we will take, for every generation, the mean of these values on all the series. In this way we can reduce the sampling variability, different from the variability associated to the GA, and allocate a general global optimum at 1.

3.2 Time Allocation Problem

The computational time C has been fixed at 10^5 fitness evaluations and the range of allocations has been decided on the basis of R and N (the number of generations G will be derived by the constraint $C = R \times G \times N$). For the optimal number R of restarts

Table 1 Time allocations

R	G	N
1	10,000	10
5	2000	10
10	1000	10
20	500	10
1	5000	20
5	1000	20
10	500	20
20	250	20
1	2000	50
5	400	50
10	200	50
20	100	50
1	1429	70
5	286	70
10	143	70
20	72	70
1	1000	100
5	200	100
10	100	100
20	50	100

no indications are found in literature; Winker (2006) worked on a range between 1 and 50,000 with $C = 2 \times 10^7$, but the Threshold Accepting is not a population based method, so in a GA analysis we must consider a very smaller range. So, in addition to no restart ($R = 1$) we chose the values 5, 10 and 20 for R. For each of these R we considered several population sizes N: 10, 20, 50, 70, 100, to have various levels of allocations. The 20 allocations are summarized in Table 1.

Once decided the allocations we must consider the stability of the results of the study; thus we will consider 4 different generator processes:

A) $\underline{\psi} = (0.01,\ 0.15,\ 0.80)$
B) $\underline{\psi} = (0.01,\ 0.50,\ 0.35)$
C) $\underline{\psi} = (0.01,\ 0.80,\ 0.15)$
D) $\underline{\psi} = (0.01,\ 0.04,\ 0.94)$

In this way we can see how the results change overbalancing the weights of the conditional variance h_t more on the data (bigger ψ_2) or on the autoregressive term (bigger ψ_3). Experiment A and D's triplet are similar to the estimates obtained, respectively, by Bollerslev and Ghysels (1991) and Fan and Yao (2003, p. 176), based on real exchange rate data.

Figure 1 shows the histogram of QML estimates (on a logarithmic scale) of parameter ψ_3 obtained by R-package *fGarch* (Wuertz et al. 2013), using 1000 time series

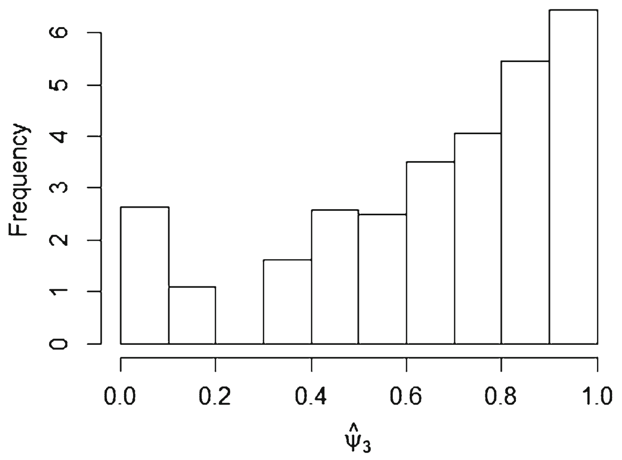

Fig. 1 QML estimates

generated according to process D: it is clear that standard numerical methods sometimes fail in providing accurate estimates.

For the present simulation study we used the open-source software R (R Core Team 2013), that was also employed to create all the graphs.

4 Results

The results of the study will be presented and discussed following two main purposes: finding GA's configurations that reach a satisfactory solution in a short time or to obtain a very high precision solution without time constraints (De Jong 1975 discussed something similar introducing the notions of off-line and on-line performance of GAs).

Figure 2 shows the progress of the best result for all allocations of the best fitness reached across generations for every experiment (A, B, C, D). It is clear that the behaviour of experiments B and C is almost identical (the curves overlap) and experiments A and D are more difficult to estimate, and this is possibly due to the large value of ψ_3 (which refers to the autoregressive term of h_t), that leads to a more complex fitness.

It is now interesting to find out, for every experiment, the overall best GA's allocation for a number of time choices, in order to gain some general indications about the association between time and GA's configurations: Table 2 summarizes these results, indicating the best $(N,\ R)$ pairs per time choice. Experiments B and C show a nearly identical progress, that dissuades from using a restart strategy and suggests an optimal population size of 50–70 chromosomes; Experiment A and D, on the other hand, seem to need higher R and/or N in the first period. As said before, these two processes have a more complex fitness, and in this case the GA possibly needs a large variety of chromosomes to gain good results in the first period, even if this means to have a small number of generations G; this aspect is regulated by both R and N, because in every generation $R \times N$ solutions are evaluated, choosing the best value among the replications. However, starting from time $t = 20,000$, all experiments conform to a

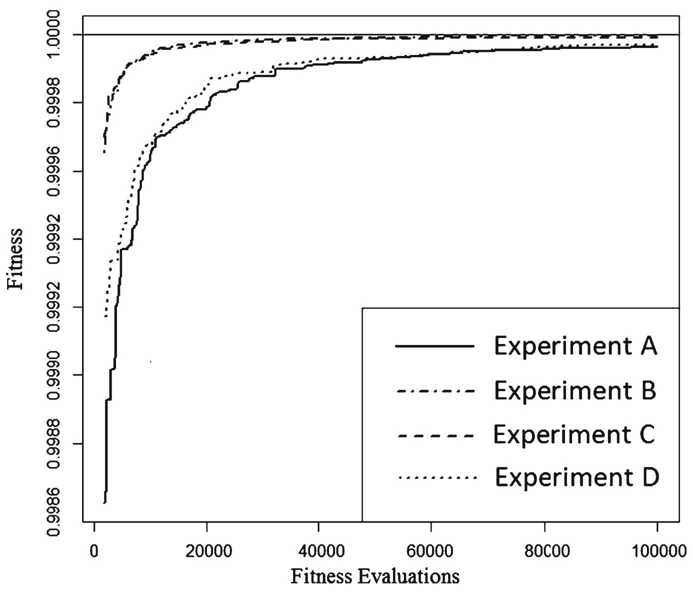

Fig. 2 Best overall fitness progress for every experiment

Table 2 Best allocations

Fitness evaluations	Exp A	Exp B	Exp C	Exp D
1000	(100, 10)	(70, 1)	(50, 1)	(50, 20)
2000	(70, 10)	(50, 1)	(50, 1)	(50, 20)
4000	(70, 10)	(50, 1)	(50, 1)	(50, 1)
6000	(100, 1)	(50, 1)	(50, 1)	(50, 1)
8000	(100, 1)	(50, 1)	(50, 1)	(50, 1)
10,000	(100, 1)	(50, 1)	(50, 1)	(50, 1)
12,000	(100, 1)	(50, 1)	(50, 1)	(50, 1)
14,000	(50, 5)	(50, 1)	(50, 1)	(50, 1)
16,000	(50, 5)	(50, 1)	(50, 1)	(50, 1)
18,000	(50, 5)	(50, 1)	(50, 1)	(50, 1)
20,000	(50, 1)	(50, 1)	(50, 1)	(50, 1)
40,000	(70, 1)	(50, 1)	(50, 1)	(50, 1)
60,000	(70, 1)	(50, 1)	(50, 1)	(50, 1)
80,000	(50, 1)	(50, 1)	(50, 1)	(50, 1)
100,000	(70, 1)	(70, 1)	(50, 1)	(50, 1)

common behaviour that allows us to derive a first conclusion: if we want to get as close as possible to the optimum without time restraints it is not convenient to restart the GA and the population size is of the order of 50 or 70.

To study the behaviour of GA's configurations in the first period we shall analyze some plots, describing the progress of the best fitness across generations conditioning on each N: in this way we can see, for each N, the behaviour of the curves of the four different restarts allocation $(1, 5, 10, 20)$. Figure 3 shows these features for Experiment A, the hardest process to estimate, according to Fig. 2. The results for Experiments B and C are similar, even if Table 2 show that the overall best short period results are obtained without restart, because of the less complex fitness. In these plots the curves are shown from time $t = R \times N$, the first time when all the individuals of the initialization (generation 0) of the GA are evaluated, to time $t = 10,000$, and the thin horizontal line represents a sub optimum, namely the 90 % of the discrepancy between a totally random initial solution of this problem and the global optimum 1. The panel for $N = 20$ suggests that the more you restart the GA the better results you will obtain; starting from $N = 50$ the behaviour of $R = 1$ becomes clear, starting from the bottom and then overtaking the curves related to higher R through time. This growth tends to become faster as N grows, because the fact that the GA doesn't get restarted, thus less chromosomes are evaluated in the first phases, is balanced up by a larger population size. We observe that also in this limited evolution runs, the best population size is intermediate, between 50 and 70. However, for the very first period, until time 2000, the $R > 1$ curves tend to overtake the $R = 1$ curve for every population size.

To synthesize our results we conducted also a logistic regression analysis, where the outcome variable is the fitness and the covariates are time t (the number of fitness evaluations, on a logarithmic scale), the number of restarts R and the population size N (both treated as qualitative factors), and their interactions. We selected, for every experiment, 50 points from all the fitness progress for every allocation, and estimated the model taking $R = 1$ and $N = 10$ categories as corner point. The model parameters estimates and the residual deviances values are summarized in Table 3. The interaction effects values are negative, though not large in absolute value, indicating that for larger N the restarts have not a positive effect on the fitness. The estimates of the coefficients of the single covariates show that population size $N = 50$ has the larger positive influence on the fitness, followed by the higher categories; the effect of the restart on the fitness, on the other hand, generally increases when the restart categories get higher. Thus the restart has a general positive effect on GA's estimation and small population sizes are not recommended.

A general indication of our results is that if speed is more important than precision, then it is more effective to generate a large quantity of initial solutions instead of spending time on generations; so in this case the total randomness gains power at the expense of GA's genetic operators. On the other hand, as we saw, if precision is the main purpose then the flow of generations becomes the most important feature of the algorithm, because the benefits of the restart mechanism in the first period are lost as time flows.

5 Conclusions and Future Work

Results from the study showed that for a GARCH(1,1) parameters estimation problem it is useful to restart the GA only when a short time is available, the population size is not very large and the main interest of the user is not the convergence to the optimum but rather the quick reaching of a satisfactory solution. On the other hand, starting from

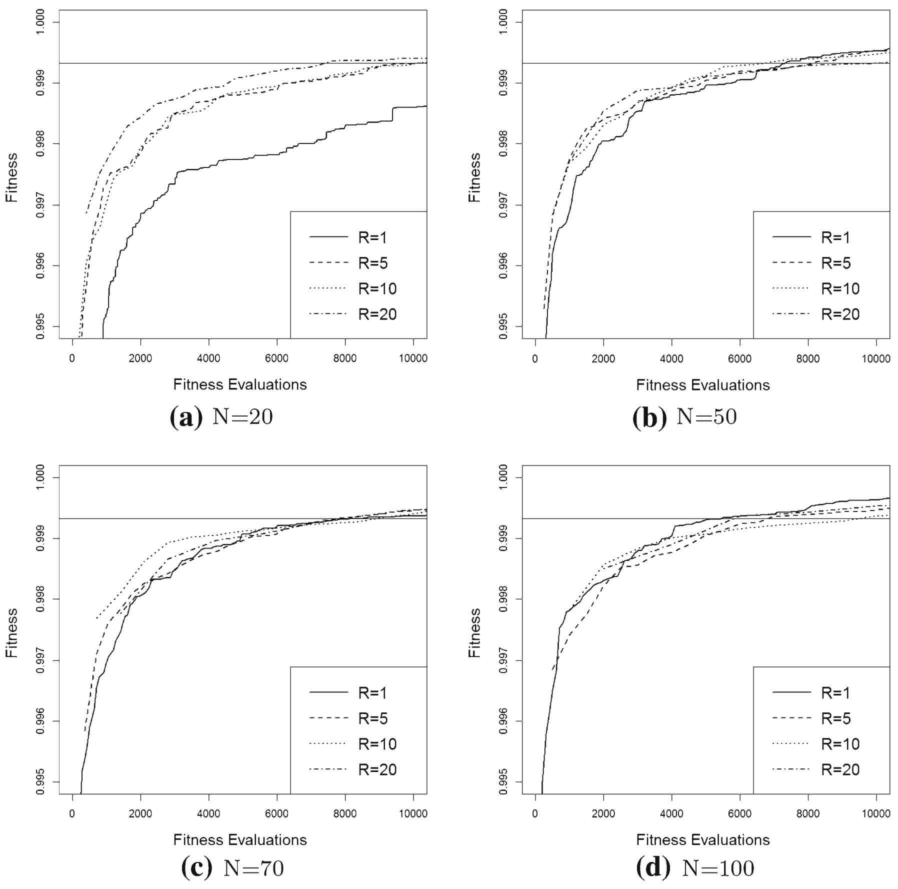

Fig. 3 Best fitness progress. **a** N = 20. **b** N = 50. **c** N = 70. **d** N = 100 for *N* fixed

a certain amount of total computational time, the restart is not convenient anymore, because the effects of the genetic operators become more important than the random nature of the restart mechanism. In fact, the restart mechanism used in the present paper allows to run parallel GAs, having a large variety of possibilities, and this is probably crucial only in the first phases of the algorithm.

A first possible direction for future works is to explore other extensions of the GAs in statistical model buildings that prevents premature convergence, for example the random immigrant scheme. Then it is possible to generalize our discussion to other models or problems where a complex fitness is implied, because it is reasonable to think that the point of time from which the restart is not useful anymore depends on the complexity of the fitness. Another important matter is represented by the mutation rate: in fact, the mutation operator was introduced to offer the opportunity of exploring different areas of the search space, and this is also what the restart mechanism does (in fact in many papers a dynamic restart mechanism has been used as a replacement for the mutation operator, see for example Ghannadian et al. 1996), so it would be

Table 3 Logistic regression model summary

	Exp A	Exp B	Exp C	Exp D
$\log(t)$	0.5048	0.6569	0.3306	0.4909
$R = 5$	0.2243	0.8658	0.2500	0.1328
$R = 10$	0.4748	1.0079	0.4284	0.3823
$R = 20$	0.8185	1.3791	0.5271	0.3807
$N = 20$	2.0151	2.5097	1.2806	1.4624
$N = 50$	2.5169	3.0442	1.4592	1.9480
$N = 70$	2.4088	2.8252	1.4421	1.8516
$N = 100$	2.4185	2.8381	1.4372	1.8352
$R = 5 * N = 20$	0.0843	−0.8073	−0.1681	0.0532
$R = 10 * N = 20$	−0.1798	−0.9139	−0.4759	−0.5012
$R = 20 * N = 20$	−0.5647	−1.5158	−0.6603	−0.3007
$R = 5 * N = 50$	−0.3033	−1.1471	−0.4468	−0.5723
$R = 10 * N = 50$	−0.6655	−1.5769	−0.8428	−1.2723
$R = 20 * N = 50$	−1.2384	−2.3673	−1.0957	−1.1532
$R = 5 * N = 70$	−0.3932	−1.2594	−0.3987	−0.5673
$R = 10 * N = 70$	−0.7257	−1.4616	−0.7713	−1.1401
$R = 20 * N = 70$	−1.3549	−2.5178	−1.0466	−1.0008
$R = 5 * N = 100$	−0.4078	−1.1531	−0.4170	−0.5099
$R = 10 * N = 100$	−0.8851	−1.4562	−0.8075	−1.0464
$R = 20 * N = 100$	−1.2511	−2.1106	−1.0814	−1.1112
Res Dev	0.0926	0.0333	0.0278	0.0385

interesting to deepen the relationship between this kind of static restart mechanism and the mutation rate. Lastly, a generalization of our discussion to other evolutionary methods may be interesting, because while several dynamic restart strategies have already been analyzed in different evolutionary fields, a static strategy like ours still needs to be deepened.

Acknowledgments The authors wish to thank Peter Winker for useful suggestions, and an anonymous referee for valuable comments.

References

Adanu, K. (2006). Optimizing the Garch model—An application of two global and two local search methods. *Computational Economics*, *28*, 277–290.

Alander, J. T. (1992). On optimal population size of genetic algorithms. In *Proceedings of CompEuro92* (pp. 65–70). Washington: IEEE Computer Society Press.

Baragona, R., Battaglia, F., & Poli, I. (2011). *Evolutionary statistical procedures—An evolutionary computation approach to statistical procedures design and applications*. Berlin: Springer.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*, 307–327.

Bollerslev, T., & Ghysels, E. (1991). Periodic autoregressive conditional heteroskedasticity. *Journal of Business & Economic Statistics*, *14*(2), 139–151.

De Jong, K. A. (1975) *An analysis of the behaviour of a class of genetic adaptive systems*. Ph.d Thesis, Dept. of Computer and Communication Sciences University of Michigan, Ann Arbor

Engle, R. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of U.K. inflation. *Econometrica*, *50*, 987–1008.

Fan, J., & Yao, Q. (2003). *Nonlinear time series: Nonparametric and parametric models*. New York: Springer.

Fukunaga, A. S. (1998). Restart scheduling for genetic algorithms. *Lecture Notes In Computer Science*, *1498*, 357–366.

Geng, L., & Zhang, Z. (2015). Forecast of stock index volatility using grey garch-type models. *The Open Cybernetics & Systemics Journal*. doi:10.2174/1874110X01509010093.

Ghannadian, F., Alford, C., & Shonkwiler, R. (1996). Application of random restart to genetic algorithms. *Intelligent Systems*, *95*, 81–102.

Goldberg, D. E. (1989). Sizing populations for serial and parallel genetic algorithms. In J. D. Schafer (Ed.), *Proceedings of the 3d conference of genetic algorithms* (pp. 70–79). San Mateo: Morgan Kaufman.

Grefenstette, J. J. (1992). Genetic algorithms for changing environments. In R. Manner & B. Manderick (Eds.), *Parallel problem solving from nature 2* (pp. 137–144). Amsterdam: Elsevier.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Hung, J. (2009). A fuzzy GARCH model applied to stock market scenario using a genetic algorithm. *Expert Systems with Applications*, *36*, 11710–11717.

Misevicius, A. (2009). Restart-based genetic algorithm for the quadratic assignment problem. In M. Bramer, F. Coenen, & M. Petridis (Eds.), *Research and development in intelligent systems XXV—proceedings of AI-2008* (pp. 91–104). London: Springer.

Phanden, R. K., Jain, A., & Verma, R. (2012). A genetic algorithm-based approach for job-shop scheduling. *Journal of Manufacturing Technology Management*, *23*(7), 937–946.

R Core Team (2013) R: A language and environment for statistical computing. Vienna: R Foundation for Statistical Computing. http://www.R-project.org/.

Rudolph, G. (1997). *Convergence properties of evolutionary algorithms*. Hamburg: Verlag Dr. Kovac.

Santamaría-Bonfil, G., Frausto-Solís, J., Vzquez-Rodarte, I., (2015). Volatility forecasting using support vector regression and a hybrid genetic algorithm. *Computational Economics*. doi:10.1007/s10614-013-9411-x.

Wang, C., & Li, G. (2001). Improving the estimations of Var-GARCH using genetic algorithm. *Journal of Systems Science and Systems Engineering*, *10*(3), 281–290.

Winker, P. (2006). The stochastic of threshold accepting: analysis of an application to the uniform design problem. In A. Rizzi & M. Vichi (Eds.), *COMPSTAT 2006—Proceeding in Computational Statistics* (pp. 495–503). Heidelberg: Physica-Verlag.

Winker, P., & Gilli, M. (2004). Applications of optimization heuristics to estimation and modelling problems. *Computational Statistics & Data Analysis*, *47*, 211–223.

Winker, P., & Maringer, D. (2009). The convergence of estimators based on heuristics: Theory and application to a GARCH model. *Computational Statistics*, *24*, 533–550.

Wuertz, D., et al. (2013). fGarch: Rmetrics—Autoregressive conditional heteroskedastic modelling. R package version 3010.82. http://CRAN.R-project.org/package=fGarch.

Zumbach, G. (2000). The pitfalls in fitting GARCH processes. In C. Dunis (Ed.), *Advances in quantitative asset management*. Amsterdam: Kluver.