World Applied Sciences Journal 14 (6): 842-849, 2011
ISSN 1818-4952

An Application of Genetic Algorithms (GA) on the IRAN Stock Market

[1]Shahram Gilaninia, [1]Mohammad Taleghani and [2]Seyyed Javad Mousavian

[1]Department of Industrial Management, Rasht Branch, Islamic Azad University, Rasht, Iran
[2]Department of Business Management, Rasht Branch, Islamic Azad University, Rasht, Iran

Abstract: This paper investigates the profitability of a simple and very common technical trading rule applied to the General Index of the Madrid Stock Market. The optimal trading rule parameter values are found using a genetic algorithm. The results suggest that, for reasonable trading costs, the technical trading rule is always superior to a risk-adjusted buy-and-hold strategy.

Key words: Technical trading rules · Genetic algorithms · Security markets

INTRODUCTION

Technical analysis is aimed at devising trading rules capable of exploiting fluctuation on the financial markets. Recent results indicate that the market timing strategy may be a viable alternative to the buy-and-hold strategy, where the assets are kept over a relatively long time period. The market timing approach is more dynamic and focuses on market fluctuations. The trading rules, through technical analysis, are devised to generate appropriate buying and selling signals.

The first results, using technical analysis in various financial domains, in the 1960s and 1970s supported the "efficient market hypothesis', which implies that there should not be any exploitable pattern in the data [1,2]. Some recent results seem to indicate otherwise [3], followed by Bessembinder and Chan [4], also demonstrated the simple trading rules could be profitable (but, without transaction costs).

Nevertheless, these developments are based on a priori rules determined through technical analysis. The emergence of new technology, in particular evolutionary algorithms, allows a system to automatically generate and adapt trading rules to particular applications. Genetic algorithms [5] have already been applied to a number of financial applications [6]. For learning trading rules, the genetic programming (GP) approach of Koza [7] looks more promising, as it provides a flexible framework for adjusting the trading rules. Although, the first attempts by Chen and Yeh [8] and Allen and Karjalainen [8] on the stock exchange markets did not show any excess returns with regard to the buy and hold approach, other recent applications of GP are more encouraging [10-12].

A considerable amount of work has provided support for the view that simple technical trading rules (TTRs) are capable of producing valuable economic signals [3,4,13,14]. However, the majority of these studies have ignored the issue of parameter optimization, leaving them open to the criticism of data snooping and the possibility of a survivorship bias [15, 16] respectively. To avoid this criticism, a more objective and valid approach consists in choosing TTRs based on an optimization procedure utilizing in-sample data and testing the performance of these rules out-of-sample. In this sense, a genetic algorithm is appropriate method to discover TTRs [9].

The aim of this paper is to investigate the profitability of some popular TTRs using genetic algorithm optimization procedures. Section 2 describes the TTRs examined in this paper, while Section 3 presents the genetic algorithms and. The empirical results are shown in Section 4.

Technical Trading Rules: In this section, we review the different ingredients that constitute the basis of a trading model and reformulate them in terms of simple quantities that can be used in conjunction with a genetic algorithm. We first need to specify an appropriate universe of trading rules from which the current GA may have been applied to. Real trading models can be quite complicated and may require many different rules that also depend on the models own trading history. Here we limit ourselves to simple models that depend essentially on a set of indicators that are pure functions of the price history or of the current return.

Corresponding Author: Shahram Gilaninia, Department of Industrial Management, Rasht Branh, Islamic Azad University, Rasht, Iran. Tel: +98-9113356977.

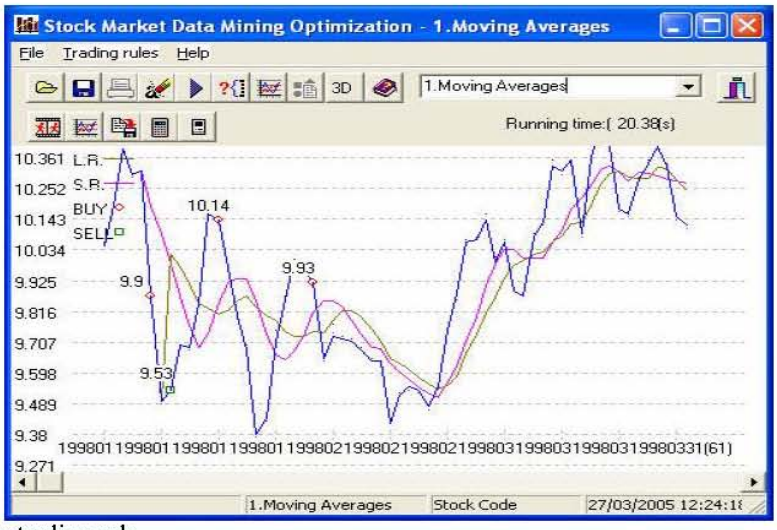

Fig. 1: Moving Average trading rule

- E.g. Moving Average rules (MA). [17]
- Define the following parameters for a simple Moving average rule:
- Short run (term/day): *sr*
- Long run (term/day): *lr*
- Short run average: *s*, is the average price of the *sr* days
- Trading price;
- Long run average: *l*, is the average price of the *lr* days
- Trading price;
- Fix band: *x*.
- Generating a buy alert signal, when *sr*-*lr*>*x*;
- Generating a sell alert signal, when *lr*-*sr*>*x*; (Figure 1)

About the data set, we divide them into two parts: in sample set (training set) and out-of-sample set (testing set). In training set, we find the robust parameters and we use the same parameters in the testing set to evaluate the result. In this paper, we use a one-year-long data as training set and continued one-year-long data as testing set. We have tested four trading rules: Filter Rules, Moving Average (MA), Support and Resistance and Channel Break-outs [17].

The evaluation used in this paper is Sharpe Ratio, which is defined by $(R_P \_ R_F) / \sigma_p$, where R_p is Expected portfolio return, R_F is Risk free rate and σ_p is portfolio standard deviation. The reason we use Sharpe Ratio is because it considers both return and risk at the same time and more and more researchers and traders are considering it.

Then, The simplest and most common trading rules are moving averages (MA). In particular, we consider a generalized MA (GMA) rule that can be represented by the following binary indicator function:

$$S(\Theta)_t = MA(\theta_1)_t - (1+(1-2\,S_{t-1})\,\theta_3)\,MA(\theta_2)_t \qquad (1)$$

Where $\Theta = [\theta_1, \theta_2\ \theta_3]$ denotes the parameters associated to the GMA rule and $MA(\theta)$ is a MA indicator defined as follows:

$$MA_t(\Theta) = \frac{1}{\theta}\sum_{i=0}^{\theta-1} P_{t-i}, \quad t = \theta, \theta+1, \ldots, N$$

The lengths of the short and long MA are given by θ_1 and θ_2 while θ_3 represents a filter parameter included to reduce the number of false buy and sell signals generated by a MA rule when price movement is nondirectional.

The GMA rule is used to indicate the trading position that should be taken at time *t*. In particular, equation (1) returns either a one o zero, corresponding to a buy or sell signal, respectively[1].

Genetic Algorithms: A genetic algorithm is a population-based search and optimization method that mimics the process of natural evolution. The two main concepts of

[1]Three different MA rules are nested within the GMA rule and can be derived individually by imposing certain restrictions on equation (1):

1) Simple MA: $\theta_1 = 1, \theta_2 > 1, \theta_3 = 0$ $S(\Theta)_t = P_t - MA(\theta_2)_t$

2) Filtered MA: $\theta_1 = 1, \theta_2 > 1, \theta_3 > 0$ $S(\Theta)_t = P_t - (1-25_{t-1})\theta_2)\,MA(\theta_2)_t$

3) Double MA: $\theta_1 = 1, \theta_2 > \theta_1, \theta_3 = 0$ $S(\Theta)_t = MA(\theta_2)_t - MA(\theta_2)_t$

natural evolution, which are natural selection and genetic dynamics, inspired the development of this method. The basic principles of this technique were first laid down by Holland and are well described, for example, in [18, 19].

The performance of a genetic algorithm, like any global optimization algorithm, depends on the mechanism for balancing the two conflicting objectives, which are exploiting the best solutions found so far and at the same time exploring the search space for promising solutions. The power of genetic algorithms comes from their ability to combine both exploration and exploitation in an optimal way [5]. However, although this optimal utilization may be theoretically true for a genetic algorithm, there are problems in practice. These arise because Holland assumed that the population size is infinite, that the fitness function accurately reflects the suitability of a solution and that the interactions between genes are very small [20].

In practice, the population size is finite, which influences the sampling ability of a genetic algorithm and as a result affects its performance. Incorporating a local search method within a genetic algorithm can help to overcome most of the obstacles that arise as a result of finite population sizes.

Incorporating a local search method can introduce new genes which can help to combat the genetic drift problem [21, 22] caused by the accumulation of stochastic errors due to finite populations. It can also accelerate the search towards the global optimum [23] which in turn can guarantee that the convergence rate is large enough to obstruct any genetic drift.

The Parallel Recombinative Simulated Annealing (PRSA) algorithm [24] fights the genetic drift problem in another way by combining the concept of the cooling schedule of simulated annealing [25] Boltzmann tournament selection [26] and standard genetic operators.

Due to its limited population size, a genetic algorithm may also sample bad representatives of good search regions and good representatives of bad regions. A local search method can ensure fair representation of the different search areas by sampling their local optima [27] which in turn can reduce the possibility of premature convergence.

In addition, a finite population can cause a genetic algorithm to produce solutions of low quality compared with the quality of solution that can be produced using local search methods. The difficulty of finding the best solution in the best found region accounts for the genetic algorithm operator's inability to make small moves in the neighborhood of current solutions [28]. Utilizing a local search method within a genetic algorithm can improve the exploiting ability of the search algorithm without limiting its exploring ability [23]. If the right balance between global exploration and local exploitation capabilities can be achieved, the algorithm can easily produce solutions with high accuracy [29].

Although genetic algorithms can rapidly locate the region in which the global optimum exists, they take a relatively long time to locate the exact local optimum in the region of convergence [30,31]. A combination of a genetic algorithm and a local search method can speed up the search to locate the exact global optimum. In such a hybrid, applying a local search to the solutions that are guided by a genetic algorithm to the most promising region can accelerate convergence to the global optimum. The time needed to reach the global optimum can be further reduced if local search methods and local knowledge are used to accelerate locating the most promising search region in addition to locating the global optimum starting within its basin of attraction.

The improper choice of control parameters is another source of the limitation of genetic algorithms in solving real-world problems [32] due to its detrimental influence on the trade-off between exploitation and exploration. Depending on these parameters the algorithm can either succeed in finding a near- optimum solution in an efficient way or fail. Choosing the correct parameter values is a time-consuming task. In addition, the use of rigid, constant control parameters is in contradiction to the evolutionary spirit of genetic algorithms [33]. For this reason, other search techniques can be utilized to set the values of these parameters whilst the search is progressing.

This process, which can be described as an automated, intelligent approach to trial and error, based on principles of natural selection, is depicted in Figure 2.

As indicated, the first step in the process is initialization, which involves choosing a population size (M), population regeneration factors and a termination criterion. The next step is to randomly generate an initial population of solutions, P(g=0), where g is the generation. If this population satisfies the termination criterion, the process stops. Otherwise, the fitness of each individual in the population is evaluated and the best solutions are "bred" with each other to form a new population, P(g+1); the poorer solutions are discarded. If the new population does not satisfy the termination criterion, the process continues.

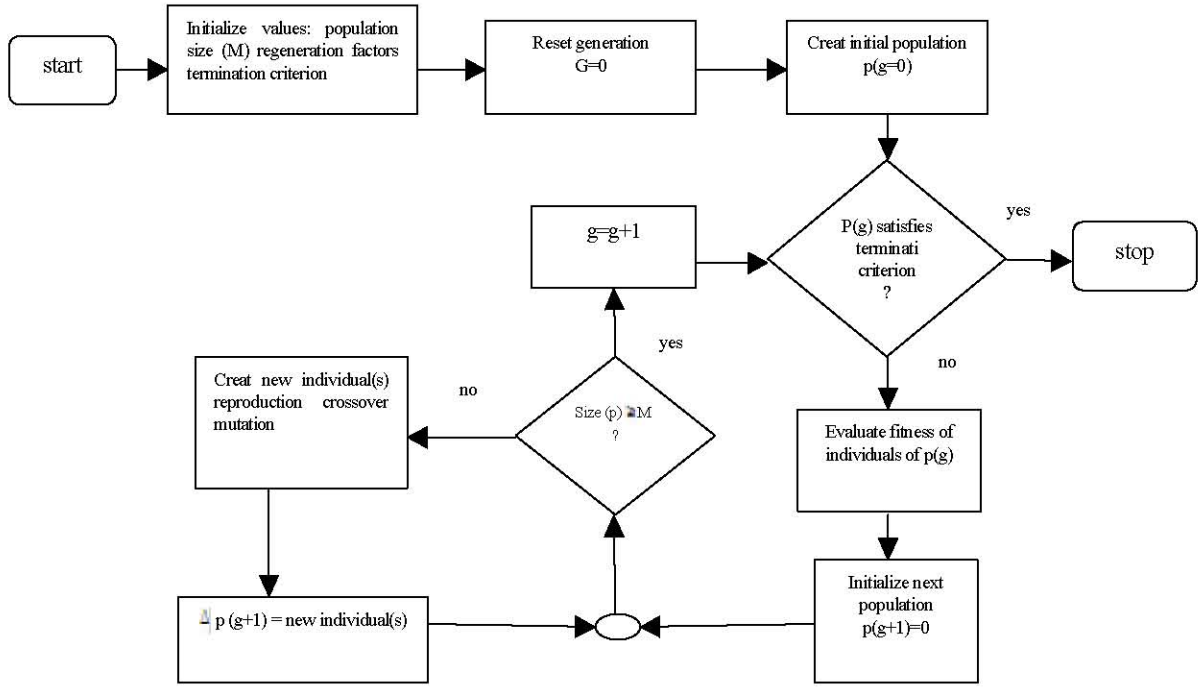

Fig. 2: The GA Process

Types of *GAs*: The simplest genetic algorithm represents each chromosome as a bit string (containing binary digits: 0s and 1s) of fixed-length. Numerical parameters can be represented by integers, though it is possible to use floating-point representations for reals. The simple GA performs crossover and mutation at the bit level for all of these.

[34, 35]. Other variants treat the chromosome as a parameter list, containing indices into an instruction table or an arbitrary data structure with pre-defined semantics, *e.g.* nodes in a linked list, hashes, or objects. Crossover and mutation are required to preserve semantics by respecting object boundaries and formal invariants for each generation can specified according to these semantics. For most data types, operators can be specialized, with differing levels of effectiveness that are generally domain-dependent. [35].

Applications: Genetic algorithms have been applied to many classification and performance tuning applications in the domain of knowledge discovery in databases (KDD). De Jong *et al.* produced *GABIL (Genetic Algorithm-Based Inductive Learning)*, one of the first general-purpose GAs for learning disjunctive normal form concepts. [36]. *GABIL* was shown to produce rules achieving validation set accuracy comparable to that of decision trees induced using *ID3* and *C4.5*. Since *GABIL*, there has been work on inducing rules [37] and decision trees [38] using evolutionary algorithms. Other representations that can be evolved using a genetic algorithm include predictors [39] and anomaly detectors [40].Unsupervised learning methodologies such as data clustering [41,42] also admit GA-based representation, with application to such current data mining problems as gene expression profiling in the domain of computational biology [43]. KDD from text corpora is another area where evolutionary algorithms have been applied [44]. GAs can be used to perform meta-learning, or higher-order learning, by extracting features [45], selecting features [46], or selecting training instances [47]. They have also been applied to combine, or fuse, classification functions [15].

Future Trends: Some limitations of GAs are that in certain situations, they are overkill compared to more straightforward optimization methods such as hill-climbing, feed forward artificial neural networks using back propagation and even simulated annealing and deterministic global search. In global optimization scenarios, GAs often manifest their strengths: efficient, parallelizable search; the ability to evolve solutions with multiple objective criteria [48]; and a characterizable and controllable process of innovation. Several current controversies arise from open research problems in GEC:

- Selection is acknowledged to be a fundamentally important genetic operator. Opinion is, however, divided over the importance of crossover verses mutation. Some argue that crossover is the most important, while mutation is only necessary to ensure that potential solutions are not lost. Others argue that crossover in a largely uniform population only serves to propagate innovations originally found by mutation and in a non-uniform population crossover is nearly always equivalent to a very large mutation (which is likely to be catastrophic).
- In the field of GEC, basic building blocks for solutions to engineering problems have primarily been characterized using schema theory, which has been critiqued as being insufficiently exact to characterize the expected convergence behavior of a GA. Proponents of schema theory have shown that it provides useful normative guidelines for design of GAs and automated control of high-level GA properties (e.g. population size, crossover parameters and selection pressure). Recent and current research in GEC relates certain evolutionary algorithms to ant colony optimization [49].

In order to determine which solution candidates are allowed to participate in the crossover and undergo possible mutation, we apply the genitor selection method proposed by Whitley [50]. This approach involves ranking all individuals according to performance and then replacing the poorly performing individuals by copies of better performing ones. In addition, we apply the commonly used single point crossover, consisting in randomly pairing candidates surviving the selection process and randomly selecting a break point at a particular position in the binary representation of each candidate. This break point is used to separate each vector into two sub vectors. The two sub vectors to the right of the break point are exchanged between the two vectors, yielding two new candidates. Finally, mutation occurs by randomly selecting a particular element in a particular vector. If the element is a one it is mutated to zero and viceversa. This occurs with a very low probability in order not to destroy promising areas of search space.

Empirical Results: The data consists of daily closing prices of the General Index of the Iran Stock Exchange and the daily 3-month rate in the interbank deposits markets, covering the 2 January 1985-15 November 2010 period (4376 observations). The total period is split into an in-sample optimization period from 2 January 1985 to 16 December 2001 and an out of- sample test period from to 16 December 2001 to 15 November 2010 (2188 observations in each sub period).

The initial population was set at 150 candidates, while the maximum number of both generation allowed and iterations without improvement was fixed at 300. The maximum the probabilities associated with the occurrence of crossover and mutation were set at 6% and 0.5%, respectively. These choices were guided by previous studies [6] and experimentation with different values. The signals from the trading rules are used to divide the total number of trading days (N) into either "in" the market (earning the market $rm_t = \ln(\frac{p_t}{p_{t-1}})$) or "out" of the market (earning the risk-free rate of return rf_t). Therefore, the objective function used to evaluate the trading rules is given by the following expression:

$$r_{tr} = \sum_{t=1}^{N} S_{t-1} rm_t + \sum_{t=1}^{N} (1 - S_{t-1}) rf_t - T * c \qquad (2)$$

Where T is the number of transactions and c is the cost per transaction. As an appropriate benchmark, we consider the return from a risk adjusted buy and hold strategy defined as

Table 1: Performance statistics

GMA trading rule						Risk-adusted buy and hold strategy			
		In-sample		Out-of-sample		In-sample		Out-of-sample	
Ttansaction Costs	Parameter Values	$\bar{r}$	*SR*	$\bar{r}$	*SR*	$\bar{r}$	*SR*	$\bar{r}$	*SR*
0.25%	(207,242,0)	33.30	0.0072	14.63	0.0068	25.36	0.0068	10.86	0.0044

Notes: GMA trading rules are identified as (*s*,*l*,*b*), where *s* and *l* are the length of the short and long period (in days) and *b* is the filter parameter. $\bar{r}$ is the average annualized return of the trading strategy and *SR* is the Sharpe ratio .

$$r_{bh} = a\sum_{t=1}^{N} rf_t + (1-a)\sum_{t=1}^{N} rm_t - 2c \quad (3)$$

where a is the proportion of trading days that the rule is out of the market. Table 1 summarizes the results. As can be seen, the best GMA rules are double MA rules, without a filter parameter (except for the case of 0 transaction costs). The Sharpe ratio and the annualized returns corresponding to the best GMA rule are higher than those from the risk adjusted buy and hold strategy, both for the in-sample and out-of-sample period4s[2]. It is interesting to note that this results holds for all transaction costs examined.

REFERENCES

1. 0Fama, E.F., 1970. Efficient capital markets: a review of theory and empirical work, J. Finance. 25: 383-417.
2. Jensen, M. and G. Bennington, 1970. Random walks and technical theories: some additional evidences, J. Finance. 25: 269-82.
3. Brock, W., J. Lakonishhock and B. LeBaron, 1992. Simple technical trading rules and the stochastic properties of stock returns, J. Finance 47: 1731-1764.
4. Bessembinder, H. and K. Chan, 1995. The profitability of technical trading rules in Asian stock markets, Pacific Basin Finance J., 3: 257-284.
5. Holland, J., 1975. Adaptation in natural and artificial systems (University of Michigan Press, Michigan).
6. Bauer, R.J., 1994. Genetic algorithms and investment strategies (John Wiley and Sons, New York).
7. **Missing.**
8. Chen, S.H. and C.H. Yeh, 1996. Toward a computable approach to the efficient market hypothesis: an application of genetic programming, J. Economic Dynamics and Contorl., 21: 1043-63.
9. Allen, F. and R. Karjalain, 1999. Using genetic algorithms to find technical trading rules, J. Financial Economics, 51: 245-271.
10. Neely, C., P. Weller and R. Dittmar, 1997. Is technical analysis in the foreign exchange market profitable? a genetic programming approach, J. Financial and Quantitative Analysis, 32: 405-26.
11. Neely, C. and P. Weller, 1999. Technical trading rules in the European monetary system, J. International Money and Finance, 18: 429-58.
12. Marney, J.P., C. Fyfe, H. Tarbert and D. Miller, 2001. Risk adjusted returns to technical trading rules: a genetic programming approach. Computing in Economics and Finance, Society for Computational Economics, Yale Univ. USA, pp: 147.
13. Mills, T., 1997. Technical analysis and the London Stock Exchange: Testing trading rules using the FT30. International J. Finance and Economics, 2: 319-331.
14. Fernández Rodríguez, F., S. Sosvilla-Rivero and J. Andrada-Felix, 1999. Technical analysis in the Madrid stock exchange, Working pp: 99- 05. FEDEA (available at ftp://ftp.fedea.es/pub/Papers/1999/DT99-05.pdf).
15. Kuncheva, L.I. and L.C. Jain, 2000. Designing classifier fusion systems by genetic algorithms. IEEE Transactions on Evolutionary Computation, 4(4): 327-336.
16. Brown, S., W. Goetzmann and S. Ross, 1995. Survival, J. Finance. 50: 853-873.
17. Ryan, S., T. Allan and W. Halbert, 1999. Data-snooping, Technical Trading Rule Performance and the Bootstrap. The J. Financial, 54(5): 1647-1692.
18. De Jong, K., 1975. "An analysis of the behavior of a class of genetic adaptive systems," Doctoral Dissertation. Ann Arbor: The University of Michigan.
19. Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning: Addison-Wesley. drift and the temporal-salience structure of problems," in 1998 IEEE International Conference on Evolutionary Computation Anchorage, USA: IEEE, pp: 535-540.
20. Beasley, D., D.R, Bull and R. Martin, 1993. "An overview of genetic algorithms: part 1, fundamentals," University Computing, 15: 58-69.
21. Asoh, H. and H. Mühlenbein, 1994. "On the mean convergence time of evolutionary algorithms without selection and mutation," in Parallel Problem Solving from Nature, PPSN III, Y. Davidor, H.-P. Schwefel and R. Manner, Eds. Berlin, Germany: Springer-Verlag, pp: 88-97.
23. Hart, W.E., 1994. "Adaptive global optimization with local search," Doctoral Dissertation. San Diego: University of California.
24. Mahfoud, S. and D. Goldberg, 1995. "Parallel recombinative simulated annealing: a genetic algorithm," Parallel Computing, 21: 11-28.

[2]The Sharpe ratio is a measure of risk-adjusted returns: $SR = \frac{r}{s\sqrt{Y}}$, where r is the average annualized return of the trading strategy, s is the standard deviation of daily trading rule returns and Y is equal to the number of trading days per year.

25. Kirkpatrick, S. Gelatt, C.D. and M.P. Vecchi, 1983. "Optimization by simulated annealing," Sci., 220: 671-680.
26. Mahfoud, S.W., 1997. "Boltzmann selection," in Handbook of Evolutionary Computation, T. Back, D.B. Fogel and Z. Michalewicz, Eds.: IOP Publising Ltd and Oxford University Press, pp: 1-4.
27. Gruau, F. and D. Whitley, 1993. "Adding learning to the cellular development of neural network: evolution and Baldwin effect," Evolutionary Computation, 1: 213-233.
28. Reeves, C., 1994. "Genetic algorithms and neighbourhood search," in Evolutionary Computing, AISB Workshop, vol. 865 Lecture Notes in Computer Science, T. C. Fogarty, Ed. Leeds, UK: Springer-Verlag, pp: 115-130.
29. Lo, A. and A. MacKinley, 1990. Data snooping biases in tests of financial assets pricing models, J. International Money and Finance. 12: 451-474.
30. Jong, K. De. 2005. "Genetic algorithms: a 30 year perspective," in Perspectives on Adaptation in Natural and Artificial Systems, L. Booker, S. Forrest, M. Mitchell and R. Riolo, Eds.: Oxford University Press.
31. Preux, P. and E.G. Talbi, 1999. "Towards hybrid evolutionary algorithms," International Transactions in Operational Res., 6: 557-570.
32. Deb, J. 1997. "Limitations of evolutionary computation methods," in Handbook of Evolutionary Computation, T. Bäck, D.B. Fogel and Z. Michalewicz, Eds.: IOP Publishing and Oxford University Press, pp: 9.
33. Eiben, A.E., R. Hinterding and Z. Michalewicz, 1999. "Parameter control in evolutionary algorithms," IEEE Transactions on Evolutionary Computation, 3: 124-141.
34. Goldberg, D.E., 2002. The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Norwell, MA: Kluwer.
35. Wikipedia. 2004. Genetic Algorithm. Available from URL: http://en.wikipedia.org/wiki/Genetic_algorithm.
36. De Jong, K.A., W.M. Spears and F.D. Gordon, 1993. Using genetic algorithms for concept learning. Machine Learning, 13: 161-188.
37. Zhou, C., W. Xiao, T.M. Tirpak and P.C. Nelson, 2003. Evolving accurate and compact classification rules with gene expression programming. IEEE Transactions on Evolutionary Computation, 7(6): 519-531.
38. Cantú-Paz, E. and C. Kamath, 2003. Inducing oblique decision trees with evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 7(1): 54-68.
39. Au, W.H., K.C.C. Chan and X. Yao, 2003. A novel evolutionary data mining algorithm with applications to churn prediction. IEEE Transactions on Evolutionary Computation, 7(6): 532-545.
40. González, F.A. and D. Dasgupta, 2003. Anomaly Detection Using Real-Valued Negative Selection. Genetic Programming and Evolvable Machines, 4(4): 383-403.
41. Hall, L.O., I.B. Ozyurt and J.C. Bezdek, 1999. Clustering with a genetically optimized approach. IEEE Transactions on Evolutionary Computation, 3(2): 103-112.
42. Lorena, L.A.N. and J.C. Furtado, 2001. Constructive Genetic Algorithm for Clustering Problems. Evolutionary Computation, 9(3): 309-328.
43. Llorà, X. and D.E. Goldberg, 2003. Bounding the effect of noise in Multiobjective Learning Classifier Systems. Evolutionary Computation, 11(3): 278-297.
44. Iba, H., 2004.Classification of Gene Expression Profile Using Combinatory Method of Evolutionary Computation and Machine Learning. Genetic Programming and Evolvable Machines, Special Issue on Biological Applications of Genetic and Evolutionary Computation (Banzhaf, W. and Foster, J. guest editors), 5(2): 145-156.
45. Atkinson-Abutridy, J., C. Mellish and S. Aitken, 2003. A semantically guided and domain-independent evolutionary model for knowledge discovery from texts. IEEE Transactions on Evolutionary Computation, 7(6): 546-560.
46. Raymer, M.L., W.F. Punch, E.D. Goodman, L.A. Kuhn and A.K. Jain, 2000. Dimensionality reduction using genetic algorithms. IEEE Transactions on Evolutionary Computation, 4(2): 164-171.
47. Cano, J.R., F. Herrera and M. Lozano, 2003. Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study. IEEE Transactions on Evolutionary Computation, 7(6): 561-575.
48. Lobo, F.G. and D. Goldberg, 1997. E "Decision making in a hybrid genetic algorithm," in IEEE International Conference on evolutionary Computation. Piscataway, USA: IEEE Press, pp: 122-125.

49. Parpinelli, R.S., H.S. Lopes and A.A. Freitas, 2002. Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computation, 6(4): 321-332.

50. Whitley, D., 1989. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best, in: D. Schaffer, ed. Proceedings of the third international conference on genetic algorithms (Morgan Kaufmann, San Mateo, California) pp: 116-121.