Evolutionary System for Generating Investment Strategies

Rafał Dreżewski and Jan Sepielak

Department of Computer Science
AGH University of Science and Technology, Kraków, Poland
drezew@agh.edu.pl

Abstract. The complexity of generating investment strategies problems makes it hard (or even impossible), in most cases, to use traditional techniques and to find the strict solution. In the paper the evolutionary system for generating investment strategies is presented. The algorithms used in the system (evolutionary algorithm, co-evolutionary algorithm, and agent-based co-evolutionary algorithm) are verified and compared on the basis of the results coming from experiments carried out with the use of real-life stock data.

1 Introduction

Investing on the stock market requires analyzing of the great number of strategies (which security should be chosen, when it should be bought or sold). Accurate analysis is important during predicting and choosing the optimal investment strategy. It plays an important role in a future success. Majority of the investment decisions are based on present and historical data. The trend anticipation depends on many assumptions, parameters and conditions. Consideration of so many assumptions, combinations of parameters and their values leads to the comparison of the great number of graphs. The evaluation of parameters of many securities is difficult and time consuming for the investor and the analyst. As a result, the investor or the analyst is able to analyze only the small subset of the possible strategies, so the optimal investment strategy is usually not found [9].

The set of the strategies which consists of indicator function is infinite because the complexity of the strategy can be unlimited. Formulas of the given strategy are functions of hundreds (or thousands) of parameters. Complexity of the problem makes it impossible to use direct search methods and instead of it a heuristic approach must be used. For example, it is possible to apply here *evolutionary algorithms* because there exist many solutions to the problem and finding optimal solutions is not necessary—suboptimal solutions are usually sufficient for an investor.

Evolutionary algorithms are optimization and search techniques, which are based on the Darwinian model of evolutionary processes [2]. One of the branches of evolutionary algorithms are *co-evolutionary algorithms* [7]. The general difference between them is the way in which the fitness of the individual is evaluated. In the case of evolutionary algorithms the fitness of the individual depends only on how "good" is the solution of the given problem encoded within its genotype. In the case of co-evolutionary algorithms the fitness of the individual depends on the values of other individuals' fitness. The value of fitness is usually based on the results of tournaments, in which the given individual and some other individuals from the population are engaged. Co-evolutionary algorithms are generally applicable in the cases in which it is difficult or even impossible to formulate explicit fitness function. Co-evolutionary interactions between individuals have also other positive effects on the population—for example maintaining population diversity, which is one of the biggest problems in the case of "classical" evolutionary algorithms.

Agent-based (co-)evolutionary algorithms are the result of research on decentralized models of evolutionary computations. The basic idea of such approach is the realization of evolutionary processes in multi-agent system, which leads to very interesting class of systems: (co-)evolutionary multi-agent systems—(Co)EMAS [3]. Such systems have some features which radically differ them from "classical" evolutionary algorithms. The most important of them are the following: synchronization constraints of the computations are relaxed because the evolutionary processes are decentralized (individuals are agents), there exist the possibility of constructing hybrid systems using many different computational intelligence techniques within single, coherent agent architecture, there are possibilities of introducing new evolutionary and social mechanisms, which were hard or even impossible to introduce in the case of classical evolutionary algorithms. (Co)EMAS systems have been already applied to multi-modal optimization and multi-objective optimization problems. Another area of applications is the modeling and simulation of social and economical phenomena.

In the paper the component system for generating investment strategies is presented. In the system three algorithms were implemented: "classical" evolutionary algorithm, co-evolutionary algorithm, and agent-based co-evolutionary algorithm. These algorithms were compared during the series of experiments, which results conclude the paper.

2 Previous Research on Evolutionary Algorithms for Generating Investment Strategies

During last years there can be observed the growing interest in applying biologically inspired algorithms to economic and financial problems. Below, only selected applications of evolutionary algorithms in systems supporting investment decision making are presented.

S. K. Kassicieh, T. L. Paez and G. Vora used the genetic algorithm for supporting the investment decisions making [5]. Their algorithm operated on historical stock data. The tasks of the algorithm included selecting company to invest in. The time series of the considered companies were given. In their system some logical operations were carried out on the data. The genetic algorithm was used to determine, which logical operators should be applied in a given situation.

O. V. Pictet, M. M. Dacorogna, R. D. Dave, B. Chopard, R. Schirru and M. Tomassini ([6]) presented the genetic algorithm for the automatic generation of trade models represented by financial indicators. Three algorithms were implemented: genetic algorithm (it converged to local minima and had the poor capability of generalization), genetic algorithm with fitness sharing technique developed by X. Yin and N. Germay [10] (it explored the search space more effectively and had more ability to find diverse optima), and genetic algorithm with fitness sharing technique developed by authors themselves in order to prevent the concentration of the individuals around "peaks" of fitness function (it had the best capability of generalization). The proposed algorithms selected parameters for indicators and combined them to create new, more complex ones.

F. Allen and R. Karjalainen ([1]) used genetic algorithm for finding trading rules for S&P 500 index. Their algorithm could select the structures and parameters for rules. Each rule was composed of a function organized into a tree and a returned value (signal), which indicated whether stocks should be bought or sold at a given price. Components of the rules were the following: functions which operated on historical data, numerical or logical constants, logical functions which allowed for combining individual blocks in order to build more complicated rules. Function in the root always returned logical value, which ensured the correctness of the strategy. Fitness measure was based on excess return from the buy-and-hold strategy, however the return did not excess the transaction cost.

3 Evolutionary System for Generating Investment Strategies

In this section the architecture of the component system for generating investment strategies is presented. Also, the three algorithms (evolutionary algorithm, co-evolutionary algorithm, and agent-based co-evolutionary algorithm) used as computational components are described.

3.1 The Architecture of the System

Fig. 1 shows the system's architecture model. The system has the following basic components:

– *DataSource*—supplies the data to the strategy generator. Historical sessions' stock data are used.

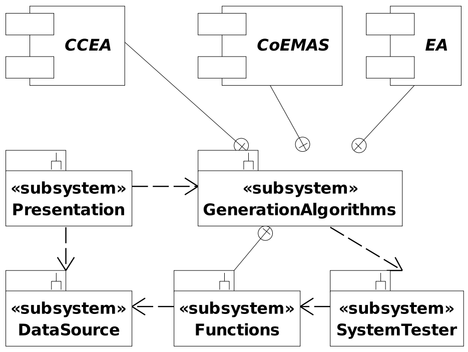

Fig. 1. The architecture of the system used in experiments

- *Functions*—contains all classes, which are necessary for creating formulas of strategies. It allows to carry out basic operations on formulas such as: initialization, exchange of single functions, adding new functions or removing existing ones. Formulas can be tested on data and, in such a case, the results will be returned.
- *SystemTester*—allows to test strategies. It can prepare reports concerning the transactions and containing the information about the gained profit. It is used by the generation algorithms to estimate the fitness.
- *GenerationAlgorithms*—contains implementation of three algorithms which generate strategies: evolutionary (EA), co-evolutionary (CCEA) and co-evolutionary multi-agent (CoEMAS). This component includes the mutation and recombination operators and also the fitness estimation mechanisms.
- *Presentation*—contains the definitions of GUI forms, which are used to monitor the generation algorithms and present the results.

3.2 The Algorithms

In all three algorithms the strategy is a pair of formulas. First formula indicates when one should enter the market and the second indicates when one should exit the market. Each formula, which is a part of strategy can by represented as a tree, which leaves and nodes are functions performing some operations. Such tree has a root and any quantity of child nodes. The function placed in the root always returns logical value. Fig. 2 shows the tree of the formula, which can be symbolically written in the following way: $STE(WillR(20), 30) > 10.0$.

Formula tree is represented in memory as a tree. The root node is an object which contains the references to the functions and the references to parameters. These parameters are also the same objects as the root. The leaves of the tree are the objects which do not contain parameters. When formula is executed recursive calls occur. In the beginning, the root requires values of all parameters needed to invoke its function. Then the control flows to objects of the parameters. The parameters objects behave in the same way as the root object. Leaves do not contain parameters, so they can return the value required by the parent node.

Functions (which formulas are composed of) were divided into four categories:

- Functions returning stock data, e.g. Close (returns close prices), Open (returns opening prices), Volume (returns volumes). There are 6 such functions.

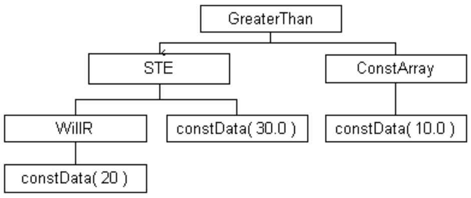

Fig. 2. Tree of exemplary formula

- Mathematical functions, e.g. Sin (sine function), Add (the addition), Pow (the power function). There are 40 such functions.
- Tool functions, e.g. Cross (finds a cross point of two functions), STE (calculates standard error), Outside (finds outside days). There are 33 of them.
- Indicator functions, e.g. AD (calculates the Accumulation/Distribution indicator), MOV (calculates diverse moving averages), WillR (calculates Williams' %R indicator). There are 14 of them.

There are 93 functions altogether. These functions accept the following types of parameters: constants (integer, float or enum), array of constant float values, and values returned by other function (array of logical values or array of float values). Logical constant does not exist because it was not needed for formula building.

Evolutionary Algorithm (EA). In the evolutionary algorithm genetic programming was used. The genotype of individuals simultaneously contains formula for entering and exiting the market. Estimation of the fitness of the strategies is carried out on the loaded historical data. Two tables of logical values are created as a result of execution of the formulas. The first one relates to the purchase action (entering the market) and the second one relates to the sale action (exiting the market). Profit computing algorithm processes tables and determines when a purchase and a sale occur. Entering the market occurs when a system is outside the market and there is the value of "true" in the enter table. Exiting the market occurs when the system is on the market and there is value of "false" in the exit table. In the other cases no operation is performed. The profit/loss from the given transaction is estimated when exiting the market and it is accumulated. The cost of each transaction is included—the commission is calculated by subtracting a certain constant from the transaction value.

Apart from the profit/loss there are also other criteria, which are included in the fitness estimation. The first one is the formula complexity—too complex formulas can slow down the computations and increase the memory usage. The complexity of the formulas is determined through the summing up of all component functions. The second criteria is the length of the transaction—it depends on the preference of the investor.

Three kinds of recombination operator are used: *returned value* recombination, *arguments* recombination, and *function* recombination. *Returned value* recombination is performed when there are two functions with different arguments but the same returned values within the formula tree. These functions are exchanged between individuals (functions are moved with their arguments). *Arguments* recombination occurs when there are two functions with the same arguments within the parents. These arguments are exchanged between individuals during the recombination. *Function* recombination can take place when two functions have the same arguments and the same returned values.

Two types of mutation were used: *function arguments* mutation and *function* mutation. In *function argument* mutation the argument of the function must be a constant. This constant is exchanged for the other one coming from the allowed range. There are three variants of the *function* mutation:

1. Traverse on functions. If a given function should be mutated, a list of the functions which take the same argument is found. If such functions exist, the exchange is performed. If there are no such functions, mutation is not carried out.

2. A given function can be exchanged for the other, completely new one, regardless of arguments—only returned types must match. Arguments of such function are created in the random manner.
3. Similarly as in (2), but, if it is possible, parameters of the replaced function are copied to a new one.

The tournament reproduction ([2]) was used in the EA algorithm. After creating the offspring, it was added to the population of parents (the reinsertion mechanism). From such enlarged population the new base population was chosen also with the use of the tournament mechanism.

Co-Evolutionary Algorithm (CCEA). While developing the co-evolutionary algorithm for generating investment strategies, co-operative approach proposed by M. A. Potter and K. A. De Jong ([8]) was used. There are two species in the implemented algorithm: individuals representing entering the market strategies and individuals representing exiting the market strategies. Interaction between these species relies on co-operation. During the fitness estimation process individuals are selected into pairs, which form the complete solution. In the first generation, for each evaluated individual from the first species a partner for co-operation from the second species is chosen randomly. For the complete solution created in this way, the fitness is computed and assigned to the individual that is being evaluated. In the next generations, the best individual from the opposite species is chosen for the evaluated individual. The pairs of individuals gaining the best profit are the result of the algorithm.

Co-Evolutionary Multi-Agent System (CoEMAS). Co-evolutionary multi-agent system used in the experiments is the agent-based realization of the co-evolutionary algorithm. Its general principles of functioning are in accordance with the general model of co-evolution in multi-agent system [3]. The CoEMAS system is composed of the environment (which include computational nodes—islands—connected with paths) and agents, which can migrate within the environment. The selection mechanisms is based on the resources, which are defined in the system. The general rule is such, that in the each time step the environment gives more resources to "better" agents and less resources to "worse" agents. The agents use the resources for every activity, like migration, reproduction, and so on. Each time step, individuals lose some constant amount of the possessed resources (the agents can live more than one generation), which is given back to the environment. The agents make all their decisions independently—especially those concerning reproduction and migration. They can also communicate with each other and observe the environment.

In the CoEMAS algorithm realized in the system for generating investment strategies each co-evolutionary algorithm (CCEA) is an agent which is located on one of the islands and independently carries out the computations. The population of each co-evolutionary algorithm also consists of the agents (there are two species of agents within each population, like in the case of CCEA).

The method of the reinsertion is different than the one used in the previous algorithms, and now works on the basis of resources. The agent which has the greater amount of resource wins the tournament. Also, on the basis of the amount of the possessed resource each individual decides whether it is ready for the reproduction. During the recombination parents give the offspring certain quantity of their resources. The possibility of migration of the agent-individual from one population to another was added as well.

4 The Experiments

In order to examine the generalization capabilities of the system and compare the proposed algorithms, strategies which earn the largest profit per year for random stocks were sought. An attempt was made to determine, which algorithm generalizes in a best way and what quantity of stocks should be used for strategies generation so that the system would not overfit. It is also interesting which algorithm generates the best strategies and has the smallest convergence.

4.1 Plan of the Experiments

The presented results of the experiments comparing the quality of the generated strategies and convergence properties of the considered algorithms are average values from 30 runs of the algorithms. Each algorithm was run for 500 generations on the data of 10 randomly chosen stocks. The session stock data came from the WIG index ([4]) and the period of 5 years was chosen (from 2001-09-29 to 2006-09-29). The size of the population in all the algorithms was equal to about 40 individuals (CoEMAS approach uses variable-size populations).

All experiments were made with the use of optimal values of the parameters. These values were found during consecutive experiments. The algorithms were run 10 times for each parameter value coming from the established range and average results were computed—on this basis the set of best parameters' values was chosen.

While examining the generalization capabilities, each algorithm generated the solution for n random stocks (stage 1). Next, different n stocks were chosen ten times at random and the profit was calculated using the best strategy obtained in the stage 1. Then, the average of these profits was counted. These calculations were carried out four times for $n = 2,5,7,10$.

Like in the first type of experiments (when the quality of the solutions and the convergence properties were compared), populations had similar sizes in the case of all three algorithms. All experiments were carried out on the machine with one AMD Sempron 2600+ processor.

4.2 Results

Fig. 3 shows the average fitness (from 30 experiments) of the best individuals for each generation. Presented results show that the evolutionary algorithm achieved the best results. The co-evolutionary algorithm achieved a little bit worse results. The quality of the solution generated by the CoEMAS was close to that of the CCEA.

Fig. 4 presents the plot of the convergence. The convergence is the phenomenon of losing by the evolutionary algorithm the ability to search the solution spaces before finding a solution which would be a global optimum. It is manifested by the occurrence

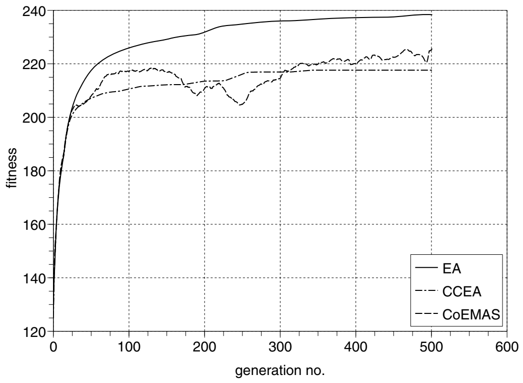

Fig. 3. Fitness of the best individuals (average values from 30 experiments)

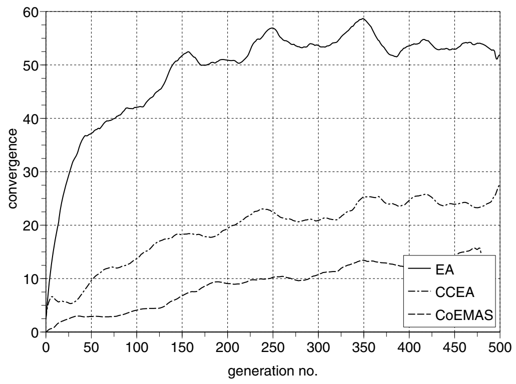

Fig. 4. Convergence values for three compared algorithms (average from 30 experiments)

of the pairs composed of identical individuals in the population. In the case of convergence the evolutionary algorithm had the worst results. Large convergence occurred already at the beginning and from 200 generation it was in the range from 50% to 60%. The co-evolutionary algorithm appeared to be much better. CoEMAS had the smallest convergence. For the last two algorithms convergence grew slowly from the beginning, but even in the end of the experiment it did not rise much.

In the table 1 the results of experiments, which goal was to investigate the capability of generalization of all compared algorithms are presented. The results show that while generating a strategy, at least 7 stocks should be used, so that the strategy could be used on any stock and earn profits in any situation (see tab. 1). If there are more stocks used during the strategy generation, the profit will be greater in the case of random stocks. For random stocks, when there was 3 or 5 of them in the group, the profits are varied and unstable. For this reason it is difficult to compare implemented algorithms with *buy and hold* strategy. It is not so, when the number of stocks in the group is 7 or 10. In the case of the random stocks, buy and hold strategy was always better (on average 2.67

Table 1. Generalization capabilities of the algorithms

Algorithm	No. of stocks in the group	Profit (%) per year (stocks from the group)	Profit (%) per year for buy & hold strategy (stocks from the group)	Profit (%) per year (random stocks)	Profit (%) per year for buy & hold strategy (random stocks)
EA	3	133.24	86.24	8.79	34.81
	5	89.15	28.85	1.8	33.62
	7	68.32	21.43	13.74	23.72
	10	87.57	24.29	20.81	26.07
CCEA	3	115.63	46.12	3.29	21.07
	5	64.82	8.83	7.39	31.36
	7	85.75	45.24	14.69	24.64
	10	69.06	26.37	20.75	25.14
CoEMAS	3	87.39	12.64	3.97	30.28
	5	66.97	7.64	-1.93	20.72
	7	86.93	39.48	19.01	34.47
	10	46.67	18.67	24.6	24.98

times) than the strategies generated by all three evolutionary algorithms, but for the stocks from the learning set generated strategies were always better (on average 1.45 times) than buy and hold strategy.

5 Summary and Conclusions

Generating investment strategies is generally very hard problem because there exist many assumptions, parameters, conditions and objectives which should be taken into consideration. In the case of such problems finding the strict solution is impossible in most cases and sub-optimal solution is usually quite sufficient for the decision maker. In such cases some (meta-)heuristic algorithms like biologically inspired techniques and methods can be used. In this paper the system for generating investment strategies, which uses three types of evolutionary algorithms was presented. The system can generate strategies with the use of "classical" evolutionary algorithm, co-evolutionary algorithm, and agent-based co-evolutionary algorithm. These algorithms were verified and compared with the use of real-life data coming from the WIG index.

The presented results show that evolutionary algorithm generated the individual (strategy) with the best fitness, the second was agent-based co-evolutionary algorithm, and the third co-evolutionary algorithm. When the population diversity (convergence) is taken into consideration, the results are quite opposite: the best was CoEMAS, the second CCEA, and the worst results were reported in the case of EA. Such observations generally confirm that co-evolutionary and agent-based co-evolutionary algorithms maintain population diversity much better than "classical" evolutionary algorithms. This can lead to stronger abilities of the population to "escape" from the local minima in the case of highly multi-modal problems.

When we consider the generalization capabilities (profit gained from 7 and 10 random stocks during one year) of the strategies generated with the use of each evolutionary algorithm, the best results were obtained by CoEMAS (21.8% profit on the average), the second was CCEA (on the average 17.7%), and the worst results were obtained in the case of EA (17.3% on the average). Implemented algorithms provide better results than buy and hold strategy for stocks from the learning set and worse results in the case of the random stocks.

The future research could concentrate on additional verification of the proposed algorithms, and on the implementation and testing of other co-evolutionary mechanisms—especially in the case of the most promising technique (CoEMAS).

References

1. Allen, F., Karjalainen, R.: Using genetic algorithms to find technical trading rules. Journal of Financial Economics 51(2), 245–271 (1999)
2. Bäck, T., Fogel, D., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. IOP Publishing and Oxford University Press, Oxford (1997)
3. Dreżewski, R.: A model of co-evolution in multi-agent system. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, Springer, Heidelberg (2003)
4. Historical stock data, `http://www.parkiet.com/dane/dane_atxt.jsp`
5. Kassicieh, S.K., Paez, T.L., Vora, G.: Investment decisions using genetic algorithms. In: Proceedings of the 30th Hawaii International Conference on System Sciences, IEEE Computer Society, Los Alamitos (1997)
6. Pictet, O.V., et al.: Genetic algorithms with collective sharing for robust optimization in financial applications. Technical Report OVP.1995-02-06, Olsen & Associates (1995)
7. Paredis, J.: Coevolutionary algorithms. In: Bäck, T., Fogel, D., Michalewicz, Z. (eds.) Handbook of Evolutionary Computation, 1st supplement, IOP Publishing and Oxford University Press, Oxford (1998)
8. Potter, M.A., De Jong, K.A.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation 8(1), 1–29 (2000)
9. Tertitski, L.M., Goder, A.G.: Method and system for visual analysis of investment strategies. US Patent 6493681 (December 2002)
10. Yin, X.: A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In: Forrest, S. (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco (1993)