QUANTITATIVE METHODS IN ECONOMICS
Volume XXI, No. 4, 2020, pp. 235 – 241

Received: 02.12.2020
Accepted: 23.12.2020

SIMULATION OF PARTICLE SWARM OPTIMIZATION FOR INVESTMENTS ON STOCK MARKET

Maciej Janowicz https://orcid.org/0000-0002-1584-2089
Institute of Information Technology
Warsaw University of Life Sciences – SGGW
e-mail: maciej_janowicz@sggw.edu.pl
Andrzej Zembrzuski https://orcid.org/0000-0001-5943-3968
Institute of Information Technology
Warsaw University of Life Sciences – SGGW
e-mail: andrzej_zembrzuski@sggw.edu.pl

Abstract: This work reports simulations performed using Particle Swarm Optimization (PSO) as applied to investments on the stock market. About 480 stocks belonging to *the S&P500* index have been taken into account. A naive approach has been developed in which one simulation step corresponded to one trading period. As a second ingredient of the investment strategy, the relative strength of an asset has been employed. The results are analyzed with respect to the parameters of PSO.

Keywords: particle swarm optimization; stock market

JEL classification: C53, C63

INTRODUCTION

There exists quite a widespread opinion that, with the advent of large-scale application of machine learning, especially deep learning, to the stock market analysis and investment, discretionary trading is a matter of the past. The argument goes by making a comparison with, e.g., the recent spectacular triumphs of the intelligent machine players over humans in the *Go* game [Silver et al.. 2016, Silver et al. 2018]. This conception is, however, misleading: human traders, either with their Excel charts or with their primitive, homemade random forests never have to fight against the full, enormous power of the Big-Corporate deep-learning

https://doi.org/10.22630/MIBE.2020.21.4.21

algorithms. It is discretionary traders' skills to move across the stock markets and win relatively small profits using the play of corporate giants for their advantage.

The author's personal motivation to undertake simulations has been related to the most elementary notions and problems of the reinforcement learning associated with the *N-armed bandits* [Robbins 1952]. One can easily draw a comparison between a group of "one-armed bandit" machines in a casino and the stocks on the stock market. Modified *N-armed bandits* have been applied for the construction of stock-market portfolios already in the past decade. Now, one way to proceed further would be to include a multi-agent version of the *N-armed bandit* problem. One of the possible solutions has been to employ one of many swarm algorithms. we have decided to use particle swarm optimization chiefly because of its simplicity and obvious connection with many-particle dynamical systems known from physics. Needless to say, apart from the profit itself (or related quantities like Sharpe ratio) there is nothing really to optimize, that is, there is no well-defined target function. But the way to adjust PSO to the task of finding a sub-optimal investment policy is, theoretically, rather simple. One should "only" find a suitable way to define "best particle positions" in the evolving environment of the market, from which the "best swarm position" can already be easily selected.

This work is indebted to the papers by Blackwell, Branke, and coworkers, please see, e.g., [Blackwell et al. 2002; Blackwell et al. 2004; Blackwell et al. 2006; Blackwell et al. 2008]; their work contains thorough discussions of the problems associated with dynamic optimization problems. Still, a very elementary approach has been employed here: one will not know whether and which modifications of that approach are needed in a new kind of application without having it extensively tested.

The main body of the work is organized as follows. In Section II, we describe how the simulations have been conducted. Section III contains the main results illustrated in several figures. Section IV comprises a summary and some concluding remarks.

METHODS

Instead of using a discrete version of PSO, the interval $[0,1]$ has been divided into L subintervals $\{S_k\}$ $(k = 0,1,2,\dots L-1)$, each corresponding to a stock. The Python module *yfinance* has been used to download the stock data. In the simulations, a kind of "day trading" approach has been applied. That is, it has been understood that if a particle landed in the subinterval $\{S_k\}$ corresponding to the k-th stock, the agent associated with that particle bought shares of that stock for all available cash with opening price and then sold all the shares with the closing prices. It has not been attempted to take into account spread, possible slippages or taxes.

Standard PSO equations of the form:

$$x(t+1) - x(t) = v(t)$$

$$v(t+1) - v(t) = a(t)$$

have been solved; the acceleration $a(t)$ has been given as:

$$a(t) = -\gamma v(t) + c_1 r_1 (p_{pb}(t) - x(t)) + c_2 r_2 (p_{sb}(t) - x(t)),$$

where r_1 and r_2 are random numbers drawn from the uniform distribution on the interval $[0,1]$, γ, c_1 and c_2 are matrix constants (in fact, they have been simply scalars in the present simulations. The vector p_{pb} is the "best position" of an individual particle, and p_{sb} is the "best position" of the whole swarm.

As has been stressed in the introduction, while we use PSO algorithm, we do not really solve an optimization problem. Rather, we use PSO in the spirit of N-armed bandit and attempt to find an efficient policy of buying and selling stocks. The (time-dependent) target functions is determined only implicitly via the specification of a method to obtain p_{pb} and p_{sb}.

Needless to say, PSO does not provide a trading strategy on its own. The determination of p_{pb} and p_{sb} has served as a necessary second ingredient. It has been assumed that $p_{pb}(t)$ is a vector of positions of particles with the largest relative strength of the corresponding stocks visited by the particle. That relative strength has been measured by a fictitious profit that would be achieved if a "trader" associated with the particle bought the asset at the time $t - t_0$ to sell it at the time $t - 1$. The quantity p_{sb} has been scalar because the pseudo-dynamics of the swarm has been one-dimensional. Profits have been calculated as a sum of all the profits accumulated by the "best" particle while visiting various stocks.

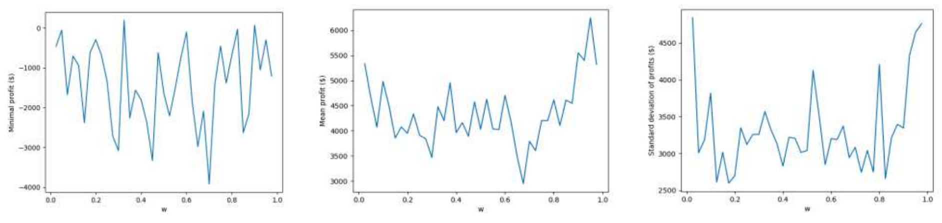
Two pairs of constants c_1, c_2 have been applied: $c_1 = c_2 = 0.5$ and $c_1 = c_2 = 1.49445$ (the latter choice is canonical). To the author's surprise, no appreciable change has been achieved when switching from the first to the second pair.

1383 trading days starting with 2015.01.01 and ending with 2020.06.30 have been taken for simulations. This has been enough to take into account both the bearish market associated with the lockdown due to the pandemia and subsequent periods of bullish developments. During that time, the value of S&P500 index has increased from 2104.50 to 3500.31, that is, by 66.325%.

RESULTS

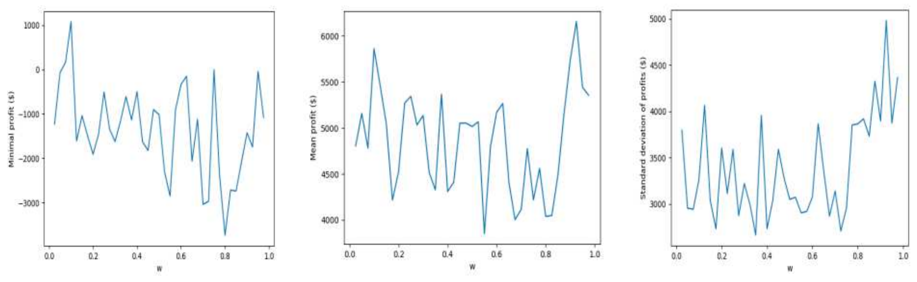
Swarms containing from 4 to 120 particles have been employed. 100 runs have been performed for every number of particles and delay parameters to obtain some reasonable statistics. This has allows us to find the mean profits and their standard deviation on averaging over the runs. In the figures below, the profits for the worst performing (that is, least profitable) runs have also been shown. In Figure 1-2 profits obtained as functions of the "inertia weight" $w = 1 - \gamma$ have been shown for the number of particles equal to 4 and 10.

Figure 1. Minimal profit (left), mean profit (center), and the standard deviation (right) of profits as functions of the "inertial weight" w for the number of particles $N = 4$, $t_0 = 2$, and $c_1 = c_2 = 0.1$



Source: own simulations

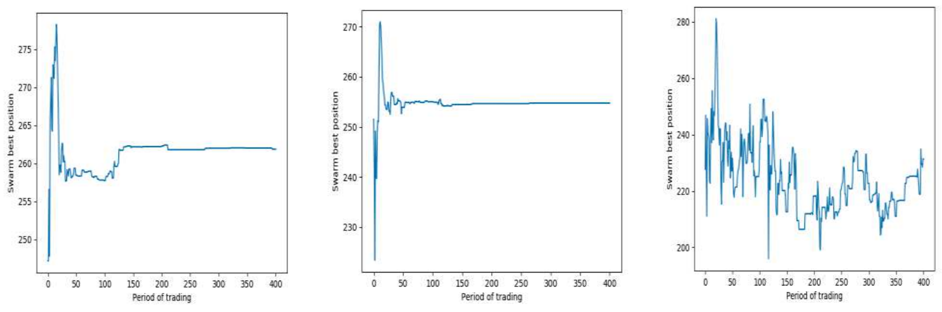
Figure 2. Minimal profit (left), mean profit (center), and the standard deviation (right) of profits as functions of the "inertial weight" w for the number of particles $N = 10$, $t_0 = 2$, and $c_1 = c_2 = 0.1$



Source: own simulations

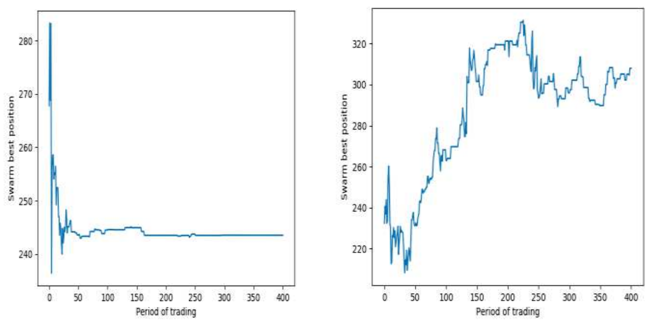
In Figures 3-4, examples of the trajectories of the best swarm positions (averaged over 100 runs) have been displayed for various number of particles and several inertial weights w.

Figure 3. Examples of trajectories of the best swarm position averagef over 100 runs for the number of particles $N = 4$, $t_0 = 2$, and $c_1 = c_2 = 0.1$. The for the "inertial weight" w has been equal to 0.1 (left), 0.5 (center) and 0.9 (right).



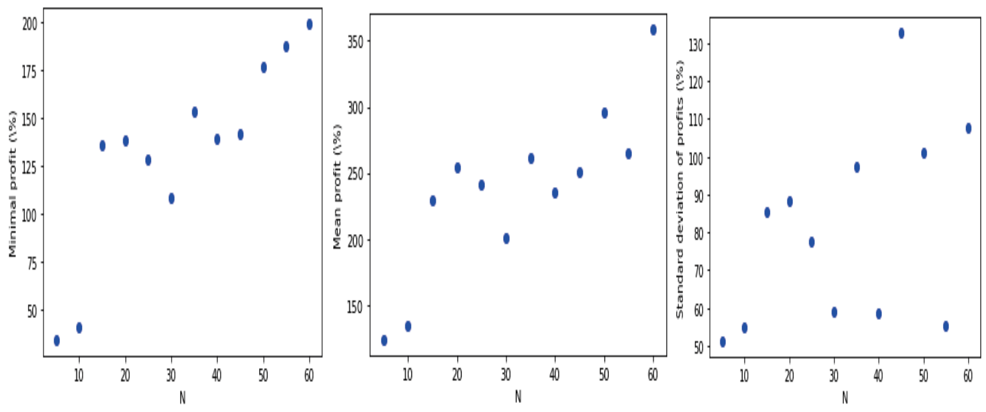
Source: own simulation

Figure 4. Examples of trajectories of the best swarm position averagef over 100 runs for the number of particles $N = 10$, $t_0 = 2$, and $c_1 = c_2 = 0.1$. The for the "inertial weight" w has been equal to 0.1 (left), 0.5 (center) and 0.9 (right).



Source: own simulation

Figure 5. The dependence of profits (in percent) on the number of particle in the sworm for $t_0 = 130$, $w = 0.5$, and $c_1 = c_2 = 2.0$: profit for the worst run (left), mean profit (center), and the standard deviations (right)



Source: own simulation

CONCLUDING REMARKS

To conclude, simulations of the behavior of particle swarms in an environment made by the stocks belonging to the $S\&P500$ index have been performed. It appears that while the method does not bring spectacular results, it does generate profits and it can serve as a valuable addition to the arsenal of traders, especially, as the author believes, individual ones. The necessary condition for such efficiency is operating with a sufficiently large number of particles in the swarm.

It appears that, if the number of particles is sufficiently large, the swarm can indeed quite efficiently find its best positon near the stocks with high relative strength. As can be seen from Figure 5, even the least profitable runs clearly outperform the benchmark if we accept it to be the value of the S&P index itself.

The results obtained here must, of course, be understood as preliminary. We have not attempted to perform any optimization of the (hyper-)parameters. Therefore, no time-dependent cross-validation has been performed. It is possible that the best parameters can be strongly market-dependent.

REFERENCES

Blackwell T. M., Bentley P. J. (2002) Dynamic Search with Charged Swarms. [in:] W. B. Langdon et al. (ed.) Proc. of the 2002 Genetic and Evolutionary Computation Conference, 19-26.

Blackwell T. M., Branke J. (2004) Multi-swarm Optimization in Dynamic Environments. [in:] G. R. Raidl (ed.) Applications of Evolutionary Computing, Lecture Notes in Computer Science, 3005, 489-500. Springer, Berlin, Germany.

Blackwell T. M., Branke J. (2006) Multi-swarms, Exclusion and Anti-convergence in Dynamic Environments. IEEE Transactions on Evolutionary Computation, 10(4), 459-472.

Blackwell T. M., Branke J., Li X. (2008) Particle Swarms for Dynamic Optimization Problems. [in:] C. Blum and D. Merkle (eds.) Swarm Intelligence, 193-217 (Springer).

Burtini G., Loeppky J., Lawrence R. (2015) Improving Online Marketing Experiments with Drifting Multi-armed Bandits. Proceedings of the 17th International Conference on Enterprise Information Systems, 2, 630-636. DOI: 10.5220/0005458706300636.

Hoffman M.D., Brochu E., de Freitas N. (2011) Portfolio Allocation for Bayesian Optimization. [in:] The Conference on Uncertainty in Artificial Intelligence.

Kennedy J., Eberhart R. C., Shi Y. (2001) Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco.

Robbins H. (1952) Some Aspects of the Sequential Design of Experiments. Bulletin of the American Mathematical Society, 58(5), 527–535, doi:10.1090/S0002-9904-1952-09620-8.

Shen W., Wang J., Jiang Y.-G., Zha H. (2015) Portfolio Choices with Orthogonal Bandit Learning. [in:] Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015).

Silver D., Huang A., Maddison C. et al. (2016) Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature 529, 484-489. https://doi.org/10.1038/nature16961

Silver D., Schrittwieser J., Simonyan K. et al. (2017) Mastering the Game of Go without Human Knowledge. Nature 550, 354-359. https://doi.org/10.1038/nature24270