One Rank Cuckoo Search Algorithm with Application to Algorithmic Trading Systems Optimization

Ahmed S. Tawfik
Department of Computer Science, Faculty of Computers and Information, Cairo University
Giza, Egypt

Amr A. Badr
Department of Computer Science, Faculty of Computers and Information, Cairo University
Giza, Egypt

Ibrahim F. Abdel-Rahman
Department of Computer Science, Faculty of Computers and Information, Cairo University
Giza, Egypt

ABSTRACT
Cuckoo search is a nature-inspired metaheuristic algorithm, based on the brood parasitism of some cuckoo species, along with Lévy flights random walks. In this paper, a modified version is proposed, where the new solutions generated from the exploration and exploitation phases are combined, evaluated and ranked together, rather than separately in the original algorithm, in addition to imposing a bound by best solutions mechanism to help improve convergence rate and performance. The proposed algorithm was tested on a set of ten standard benchmark functions, and applied to a real-world problem of algorithmic trading systems optimization in the financial markets. Experimental analysis demonstrated improved performance in almost all benchmark functions and the problem under study.

General Terms
Algorithms, Algorithmic Trading, Optimization.

Keywords
Algorithms, Algorithmic Trading, Cuckoo Search, Metaheuristics, Nature-inspired Algorithms, Optimization, Technical Analysis, Swarm Intelligence.

1. INTRODUCTION
Optimization is used everywhere, from engineering design to economics, and while resources and time are always limited, the best possible tools available are very important to be utilized efficiently. There are many algorithms which can be classified in several ways. From a simple perspective, algorithms can be classified as deterministic or stochastic. If an algorithm works in a mechanical manner, without any randomness, it is called deterministic. If there is some random nature in the algorithm, it is called stochastic, as in genetic algorithms (GA) [1] and particle swarm optimization (PSO) algorithms [2, 3]. Algorithms with stochastic components are often referred to as heuristic, or as metaheuristics in the recent literature [4–7]. Almost all modern metaheuristics algorithms have two major components of exploration and exploitation, and use a certain trade-off of randomization and local search. An example of the nature-inspired metaheuristics algorithms is the cuckoo search (CS) algorithm [7–9]. A recent study suggests that CS is potentially far more efficient than GA, PSO, and other popular algorithms [10].

Algorithmic trading is the use of a trading system with a set of rules that automatically decides, without any human discretionary intervention, on aspects of the entry and exit orders, including time, price, and quantity of the order in the financial markets [11–14]. Technical analysis is the field of security analysis concerning the evaluation of financial instruments by analyzing market data and activity, such as historical prices and volume, using charts, and other tools to identify patterns that can advise on future movements [15, 16]. Technical analysis is commonly used as the primary component in defining algorithmic trading systems rules.

This paper aims to derive a modified version of the cuckoo search algorithm and provide a comparison study of the original and proposed algorithms. The subsequent parts of this paper are organized as follows: an outline of the cuckoo search algorithm in section 2, explanation of the proposed algorithm in section 3, a performance comparison of the original and proposed algorithms in section 4, an application of algorithmic trading systems optimization in section 5, and the discussions and conclusions are presented in section 6.

2. CUCKOO SEARCH ALGORITHM
Cuckoo search is a relatively recent nature-inspired metaheuristic algorithm, developed by Xin-She Yang and Suash Deb in 2009. CS was inspired by the brood parasitism of some cuckoo bird species, in combination with the Lévy flights [17–19] random walks. Cuckoos are catching scientists' interest because of their aggressive reproduction strategy. Some species lay their eggs in communal nests of other host birds (often other species), and may remove others' eggs to increase the hatching probability of their own eggs.

CS idealized rules can be summarized as:

1. Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest.
2. The best nests with high quality of eggs will carry over to the next generations.
3. The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0,1]$, where the host bird can either throw the egg away, or abandon the nest and build a completely new nest.

These rules are used to draw the basic cuckoo search algorithm in Pseudo code 1.

While exploring new solutions, it is necessary to control the Lévy flights random walks, to avoid large moves, causing the solutions to jump outside of the search space. A step size factor that is defined according to the scale of the problem of interest should be used for this purpose. This might be an interesting subject for more research, studying the optimal utilization of the Lévy flight in optimization; for simplicity, a typical step size factor of 0.01, as suggested by the authors, is being used in this study [7, 9, 18].

Pseudo code 1. Cuckoo search algorithm

```
Objective function f(x), x = (x_1, ..., x_d);
Initialize a population of N host nests/solutions;
Define the Lévy flights step size factor s ← 0.01;

While is not stop criteria
    Find new nests, using Lévy flights as:
    For n = 1 to N (all nests)
        For d = 1 to D (all dimensions)
            Set newnest[n,d] ←
            s × NormalSample() × LévySample() ×
            (nest[n,d] − best[n,d]);
        End for
    End for

    Evaluate the new nests against the objective function
    and calculate their quality/fitness;
    Rank and keep the current best nests as:
    For n = 1 to N
        If newnest[n] is better than nest[n]
            Replace nest[n] by newnest[n]
        End if
    End for

    Replace a fraction p_a of nests as:
    Get two random permutation arrays rpx and rpy of N
    length;
    For n = 1 to N
        For d = 1 to D
            If RandomSample() ≤ p_a
                Set newnest[n,d] ←
                RandomSample() ×
                (nest[rpx[n],d] − nest[rpy[n],d]);
            End if
        End for
    End for

    Evaluate the new nests;
    Rank and keep the current best nests;

    Get the current best nest;
End while
```

3. PROPOSED ALGORITHM

The proposed algorithm, one rank cuckoo search (ORCS), applies two behavioral amendments to the original cuckoo search algorithms, in aim to improve convergence rate, and consequently achieve a better performance and accuracy. These behaviors are defined in the following subsections.

3.1 One Rank (Combined Evaluation)

The original algorithm generates new nests using Lévy flights (exploration phase) and evaluates their fitness, then replaces a fraction of nests (exploitation phase) and evaluates and ranks their fitness once more. Instead, the proposed algorithm generates new solutions using Lévy flights, replaces a fraction of them, and finally evaluates and ranks their fitness at once. This behavior allows a more conservative consumption of the function evaluations, by merging together the new solutions generated by the exploration and exploitation phases before evaluating them, and hence consuming n (population size) evaluations per iteration by the proposed algorithm against $2n$ evaluations by the original algorithm.

A one rank ratio r_{or} is initiated by 1, to allow the proposed algorithm to combine all the explorations and exploitations, until it fails to find better nests for t_{or} iterations, to trigger a gradual decrease of the one rank ratio as in Eq. 1, where t is the iteration number and D is the number of objective function dimension.

$$r_{or}^{t+1} = r_{or}^{t} \times 1 - 0.5/D. \quad (1)$$

3.2 Bound by Best Solutions

It is required to enforce the constraints defined by an objective function for all the solutions generated during the optimization process, particularly when using algorithms with some random nature components, such as the cuckoo search algorithm. The draw on Lévy flights for exploring the search space, rather than uniform random walks, tends to increase the probability for the generated solutions to get out of the defined constraints, and accordingly an increased need for a better bound behavior than a simple minimum-maximum bound.

The proposed algorithm enforces the integrity over an out of constraints solution by replacing its invalid dimensions by the corresponding dimensions drawn from randomly selected solutions among the current best solutions. A ratio of the replaced dimensions is utilizing the current best solutions found so far, and the rest is being randomly drawn by further exploring the search space. A bound by best ratio r_{bbb} is defined as in Eq. 2, and the basic bound by best solutions behavior is presented in Pseudo code 2.

$$r_{bbb} = 1 - 1/\sqrt{D}. \quad (2)$$

Pseudo code 2. Bound by best procedure

```
Objective function f(x_1, ..., x_d), x ∈ [mix, max];
Define bound by best ratio r_bbb as in Eq. 2;
For each dimension d in new solution
    If not min ≤ newsolution[d] ≤ max
        If RandomSample() ≤ r_bbb
            Select a solution randomly from the current
            N best solutions, solutions[n];
            Set newsolution[d] ← solutions[n,d];
        Else
            Set newsolution[d] ← min +
            RandomSample() × (max − min);
    End if
End for
```

3.3 One Rank Cuckoo Search Algorithm

The one rank/combined evaluation and bound by best solution functionalities, have been added to the original cuckoo search algorithm, and used to draw the basic one rank cuckoo search algorithm in Pseudo code 3.

The proposed ORCS algorithm introduced one more parameter, one rank ratio update trigger t_{or}, in addition to the two parameters employed by the original CS algorithm, population size N and abandon rate p_a. This parameter has not been meta-optimized to select the best performing setting; however the preliminary tests undergone during the algorithm development, demonstrated insensitivity in this parameter to have a major impact on the algorithm performance.

Pseudo code 3. One rank cuckoo search algorithm

Objective function $f(x), x = (x_1, \dots, x_d)$;
Initialize a population of N host nests/solutions;
Define the Lévy flights step size factor $s \leftarrow 0.01$;
Define the one rank ratio $r_{or} \leftarrow 1$;

While is not stop criteria
  Find new nests, using Lévy flights as:
  For $n = 1$ *to* N (all nests)
    For $d = 1$ *to* D (all dimensions)
      $Set\ newnest[n,d] \leftarrow$
      $s \times NormalSample() \times LévySample() \times$
      $(nest[n,d] - best[n,d])$;
    End for
  End for

  If $RandomSample() \leq r_{or}$
    Replace a fraction p_a of solutions as:
    Get two random permutation arrays rpx and rpy of N length;
    For $n = 1$ *to* N
      For $d = 1$ *to* D
        If $RandomSample() \leq p_a$
          $Set\ newnest[n,d] \leftarrow$
          $RandomSample() \times$
          $(nest[rpx[n],d] -$
          $nest[rpy[n],d])$;
        End if
      End for
    End for

    Bound by best, as in pseudo code 2;
    Evaluate the new nests;
    Rank the nests and keep the current best nests as:
    For $n = 1$ *to* N
      If $newnest[n]$ is better than $nest[n]$
        Replace $nest[n]$ by $newnest[n]$
      End if
    End for
  Else
    Bound by best;
    Evaluate the new nests;
    Rank the nests and keep the current best nests;

    Replace a fraction p_a of nests;

    Bound by best;
    Evaluate the new nests;
    Rank the nests and keep the current best nests;
  End if

  Get the current best nest;
  If no better nests since t_{or} evaluations
    Decrease r_{or}, as in Eq. 1;
  End if
End while

4. EXPERIMENTAL ANALYSIS

4.1 Benchmark Functions

The proposed algorithm was tested on 10 standard benchmark functions that are commonly used to evaluate the performance of optimization algorithms [20, 21]. The functions selected are carrying different properties of being unimodal or multimodal, separable or non separable, and having few or many local optima solutions. Table 1 lists the functions and their testing properties, and Fig. 1 shows a 3D plot of the functions in 2 dimensions.

Table 1. The benchmark functions and their testing properties

Function $f(x^{optimum}) = min_{x \in S} f(x_1, .., x_d)$	**Search Space / Optimum**
Ackley, $20 + e - 20 * e^{-0.2*\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\frac{1}{d}\sqrt{\sum_{i=1}^{d} 2\pi x_i}}$	± 32 / 0
Dixon & Price, $(x_1 - 1)^2 + \sum_{i=2}^{d} i(2x_i^2 - x_{i-1})^2 - 1$	± 10 / 0
Griewank, $\sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{1=i}^{d} \cos(\frac{x_i}{\sqrt{i}}) - 1$	± 600 / 0
Michalewicz, $\sum_{1=1}^{d} \sin(x_i) \sin(\frac{ix_i^2}{\pi})^{20}$	$\pm\pi$ / Varies by dimensions
Rastrigin, $10d + \sum_{1=1}^{d} x_i^2 - 10\cos(2\pi x_i)$	± 5.12 / 0
Rosenbrock, $\sum_{i=1}^{d-1} 100(x_1^2 - x_{i+1})^2 + (x_i - 1)^2$	± 50 / 0
Schwefel, $420.9828d - \sum_{1}^{d} x_i \sin\sqrt{\lvert x \rvert}$	± 500 / 0
Sphere, $\sum_{i=1}^{d} x_i^2$	± 100 / 0
Trid, $\sum_{i=1}^{d}(x_i - 1)^2 - \sum_{i=1}^{d} x_i - x_{i-1}$	$\pm n^2$ / Varies by dimensions
Zakharov, $\sum_{i=1}^{d} x_i^2 + (\sum_{i=1}^{d} 0.5ix_i)^2 + (\sum_{i=1}^{d} 0.5ix_i)^{24}$	$[-5, 10]$ / 0

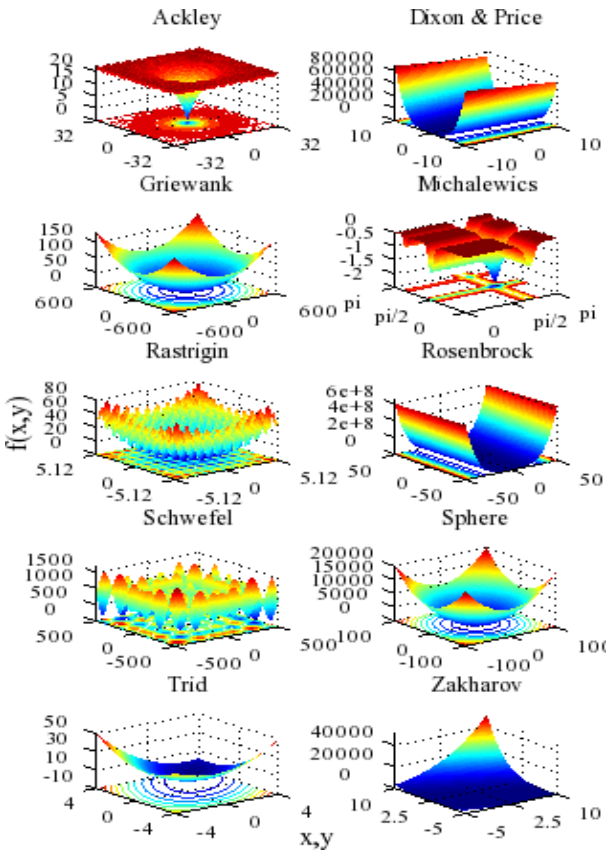

Fig. 1. A 3D plot of the benchmark functions in 2 dimensions

4.2 Comparison of CS and ORCS

The original cuckoo search (CS) algorithm with a population size $N = 15$ and discovery rate $p_a = 0.25$, as suggested by its authors, and the proposed one rank cuckoo search (ORCS) algorithm with the same settings as CS, plus a one rank ratio update trigger $t_{or} = 0.1$ (i.e. 10% of the maximum number of function evaluations) have been used for all tests.

The tests have been run on 5, 10, 25, 50, 100, and 250 dimensions for a maximum of 100000 function evaluations, with an automatic stop criteria defined as:

- Optimum fitness reached.
- All the population of N nests is holding identical eggs (i.e. all solutions' dimensions are equal).

All tests have been run for 100 times and their mean were used to carry out a meaningful statistical analysis. The comparative results of the achieved fitness are plotted in Figures 2-7.

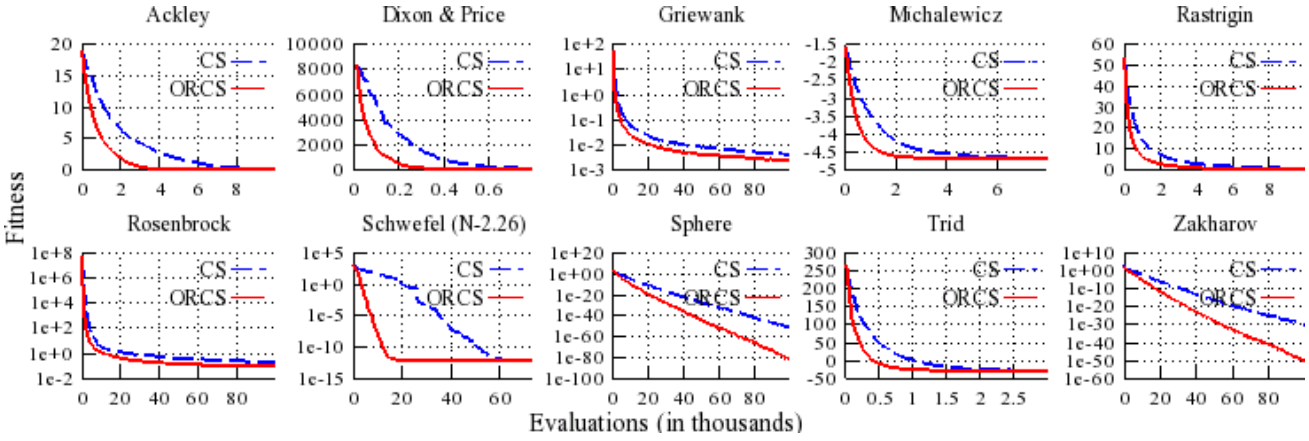

Fig. 2. Experimental results in 5 dimensions

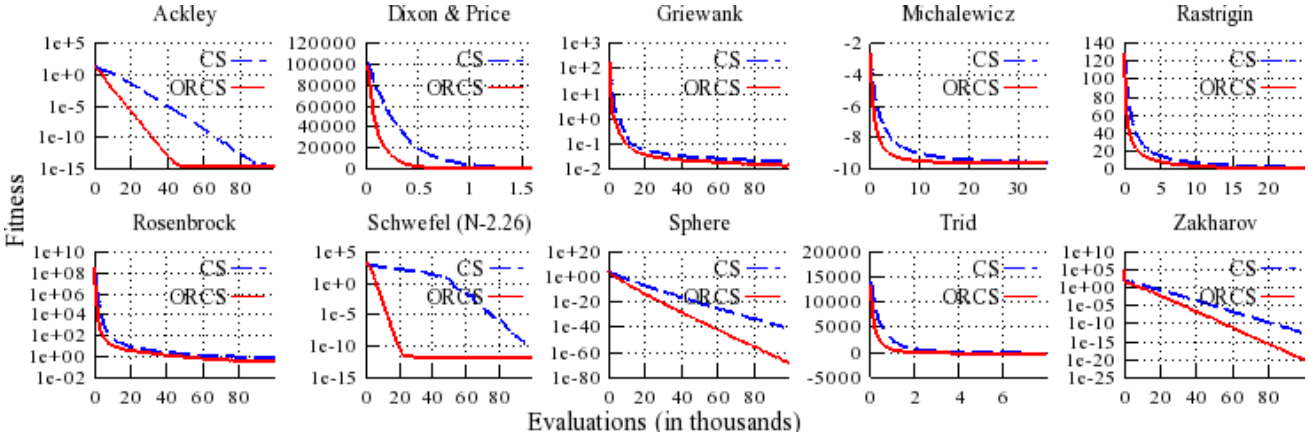

Fig. 3. Experimental results in 10 dimensions

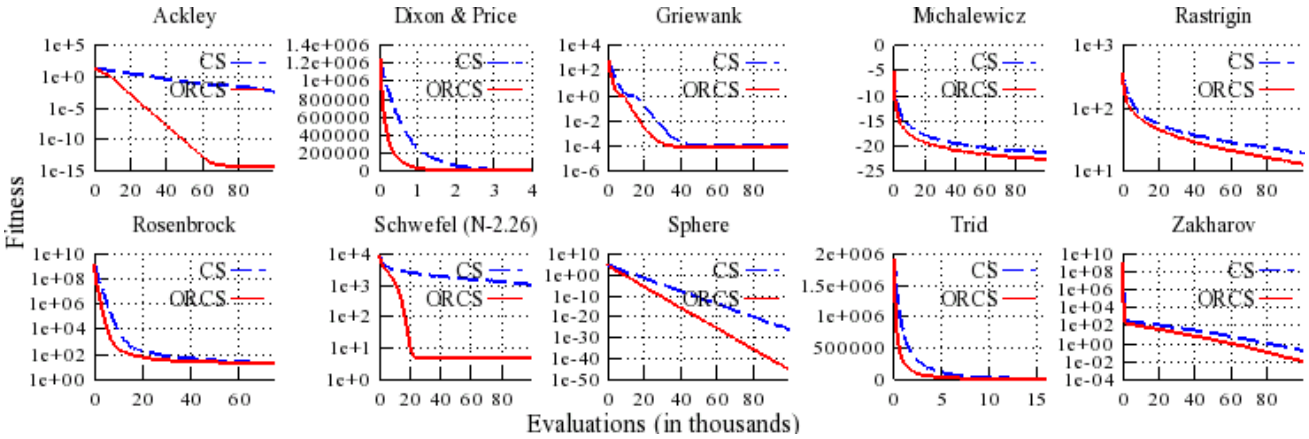

Fig. 4. Experimental results in 25 dimensions

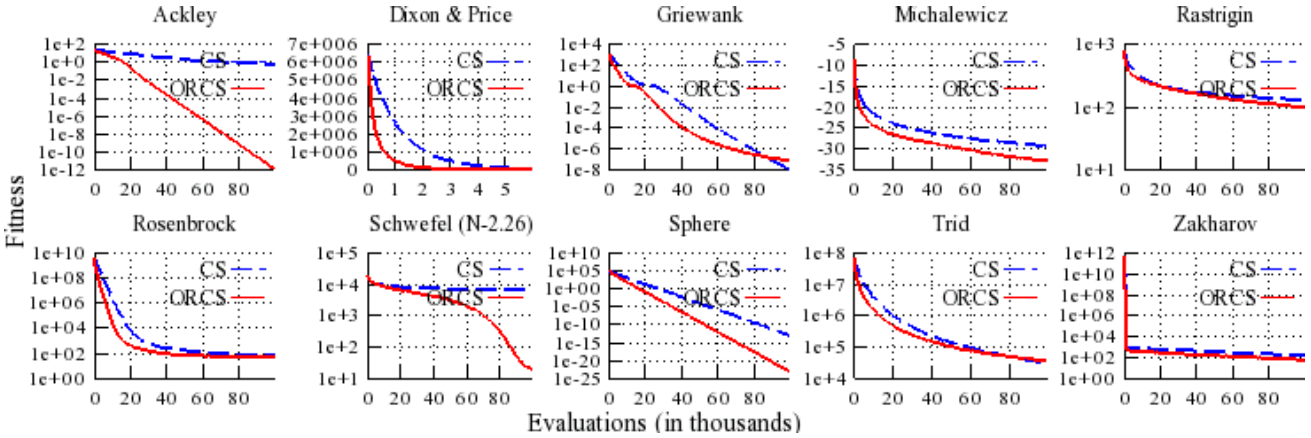

Fig. 5. Experimental results in 50 dimensions

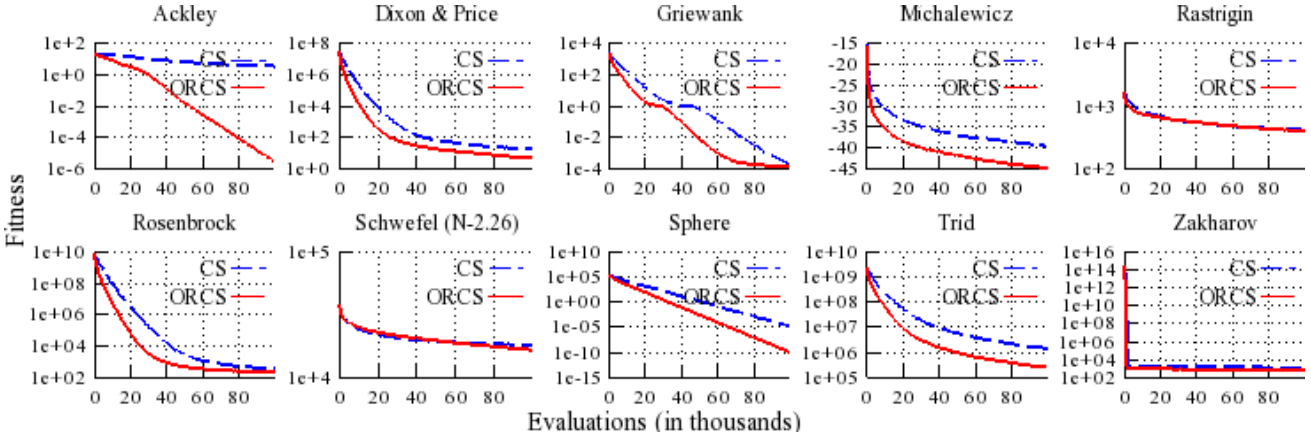

Fig. 6. Experimental results in 100 dimensions

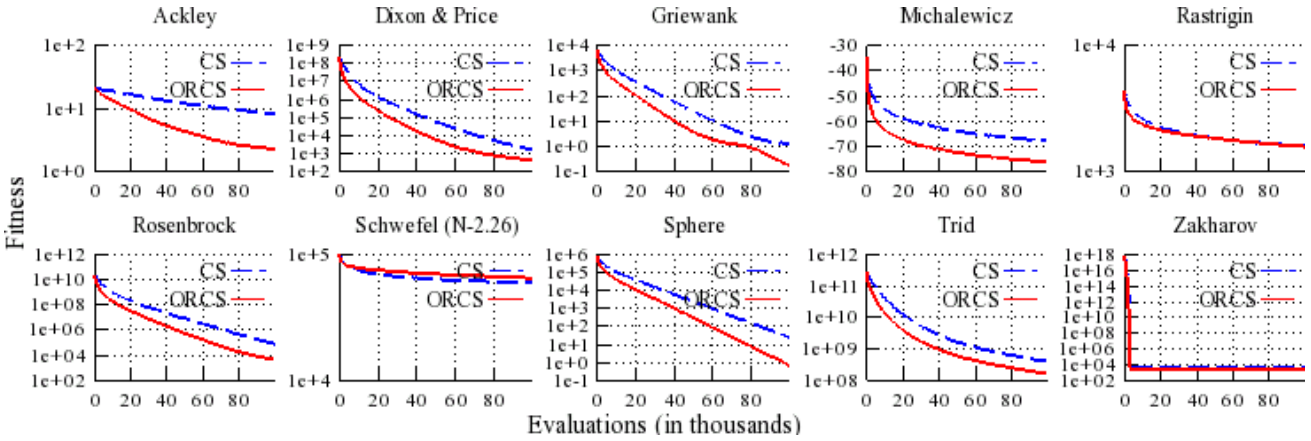

Fig. 7. Experimental results in 250 dimensions

The results obtained, from the 60 tests carried on 10 different benchmark functions and 6 different dimensions, by the CS and ORCS algorithms, suggest that ORCS outperformed CS in almost all tests. This improvement can be credited to the well balance between exploration and exploitation rules introduced by the original algorithm, backed by the conservative use of function evaluations and selective bound rules presented by the proposed algorithm.

5. APPLICATION TO ALGORITHMIC TRADING SYSTEMS OPTIMIZATION

The input parameters of a trading system are generally a mix of integer and floating point variables, and hence it can be described as a combinatorial [5] or mixed-integer nonlinear [22] optimization problem. The algorithms defined in the earlier sections need to go through an additional step before they could be successfully used to optimize a trading system with part (or all) of its input parameters are of discrete nature, such as being restricted to integer values. After generating a new nest/solution and before evaluating its fitness/performance, a round to a specific increment is

performed as in Eq. 3, where p is the solution parameter and i is the parameter increment.

$$p = Round\left(\frac{p}{i}\right) \times i. \quad (3)$$

5.1 Trading System

For the purpose of this study, an experimental trading system (MRA) has been defined, to verify and compare the efficiency of CS and ORCS algorithms in the problem of algorithmic trading systems optimization.

The main rules of the MRA trading system, responsible for generating buy and sell orders and managing positions, are built on three well-known technical analysis indicators: Moving Averages (MA), Relative Strength Index (RSI) and Average True Range (ATR) [15, 16]. The basic behavior of the MRA system is defined in Pseudo code 4.

Pseudo code 4. Experiment MRA trading system

```
Read input parameters:
StartHour, EndHour, … , RsiLevel, …,
ProtectiveStopAtrs, TrailingStopAtrs, TargetAtrs;

Define Close Moving Average (CMA), High Moving
Average (HMA) and Low Moving Average (LMA);
Define Relative Strength Index (RSI), its Trailing Level
(TL) and Slow Trailing Level (STL);
Define Average True Range (ATR);

For each price update
    If StartHour ≤ CurrentTime ≤ EndHour
        If (CMA crossed above HMA and TL > STL
        or TL crossed above STL and CMA > HMA)
        and RSI < 100 − RsiLevel
            Buy;
            Set stop loss level at:
            ProtectiveStopAtrs * ATR;
            Set profit target level at:
            TargetAtrs * ATR;
        Else if (CMA crossed below LMA and TL < STL
        or TL crossed below STL and CMA < LMA)
        and RSI > RsiLevel
            Sell short;
            Set stop loss level at:
            ProtectiveStopAtrs * ATR;
            Set profit target level at:
            TargetAtrs * ATR;
        Else if position exists
            Switch position
                Case Long:
                    Update stop loss level to:
                    HightHighSinceBuy −
                    TrailingStopAtrs * ATR;
                    Break;
                Case Short:
                    Update stop loss level to:
                    LowestLowSinceSell +
                    TrailingStopAtrs * ATR;
                    Break;
            End Switch
        End if
    End if
End For
```

5.2 Comparison of CS and ORCS

The CS and ORCS algorithms, with identical setting to those used in section 4 have been applied to optimize the MRA trading system parameters listed in Table 2.

The tests have been run on a basket of 23 instruments, selected from the foreign exchange (FOREX) market [23], for a total of 2 years of historical data, from January 2011 to December 2012.

The objective function used in the optimization of the MRA trading system is show in Eq. 4, where p^{best} is the parameters-list achieving the best performance possible, NP is the total net profit and SR is the Sharpe ratio [24] of the monthly profits.

$$f(p^{best}) = max(NP \times SR). \quad (4)$$

The tests have been run for a maximum of 10000 evaluations, with an automatic stop criterion when all the population of *N* nests is holding identical eggs.

Table 2. Experimental MRA trading system parameters

Component	Parameter	Range / Increment
Session Filter	$StartHour$	[0 , 19] / 1
	$EndHour$	[3 , 22] / 1
Order Generator (moving averages)	$ClosePeriod$	[1 , 3] / 1
	$HighLowPeriod$	[2 , 9] / 1
Order Generator (relative strength index and its trailing levels)	$RsiPeriod$	[5 , 14] / 1
	$RsiSmoothing$	[1 , 5] / 1
	$RsiAtrPeriod$	[5 , 14] / 1
	$RsiAtrSmoothing$	[5 , 14] / 1
	$TlFactor$	[0 , 2.5] / 0.1
	$StlFactor$	[0.5 , 5] / 0.1
	$RsiLevel$	[20 , 40] / 1
Position Manager (stop loss and profit target levels)	$AtrPeriod$	[5 , 24] / 1
	$ProtectiveStopAtrs$	[2 , 4] / 0.1
	$TrailingStopAtrs$	[1 , 5] / 0.1
	$TargetAtrs$	[3 , 10] / 0.1

All tests have been run for 25 times and their mean were used to carry out a meaningful statistical analysis. The comparative results of the achieved performance are plotted in Fig. 8.

It is worth mentioning that, while optimizing a trading system, it is important to avoid the overfitting of the system rules on the data being optimized, which is commonly caused by relying on a limited data or too many parameters [25–27]. This problem might be an interesting subject to study the possibilities of auto detection and prevention of overfitting on different optimization problems in general and the problem of algorithmic trading systems in specific.

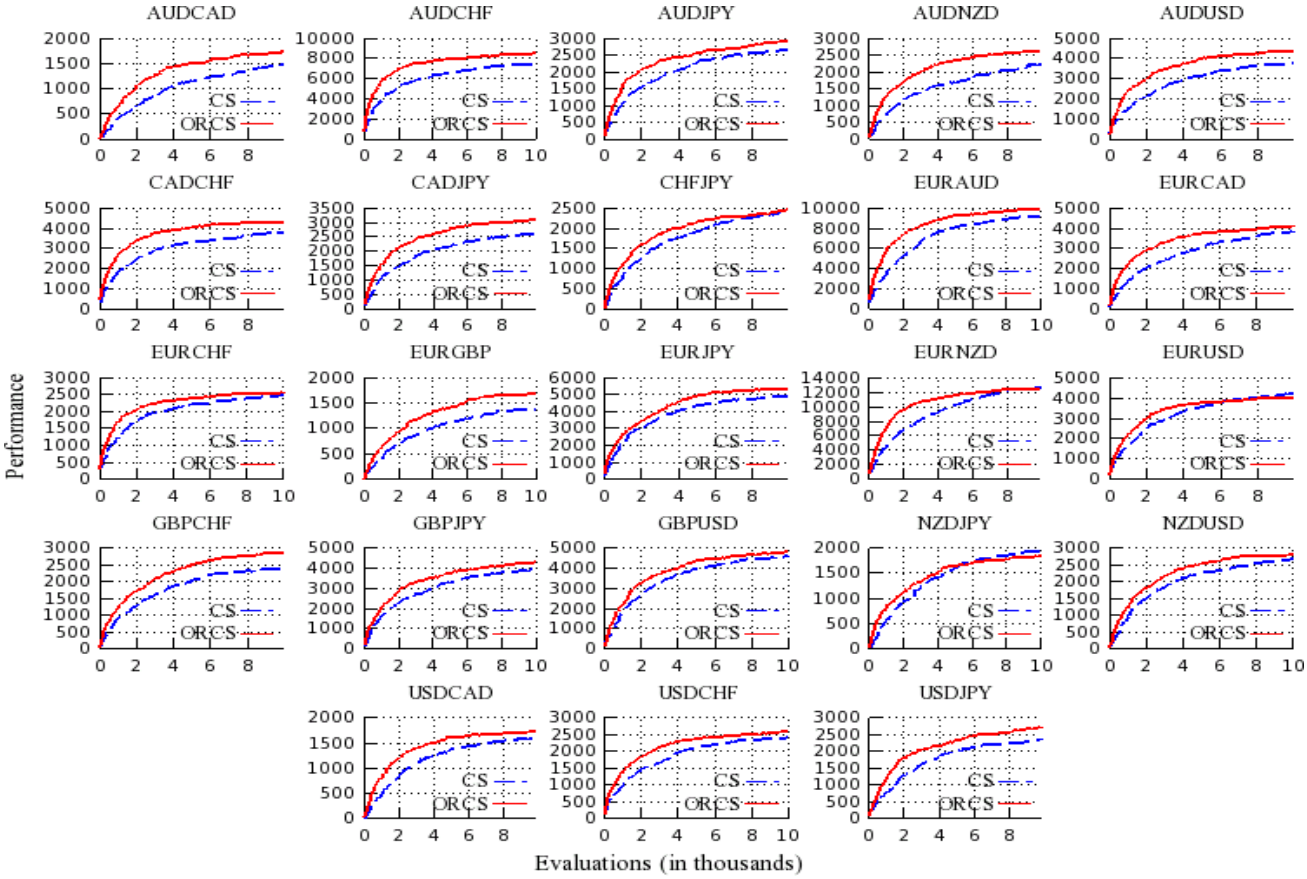

Fig. 8. Experimental results for a basket of 23 FORX instruments

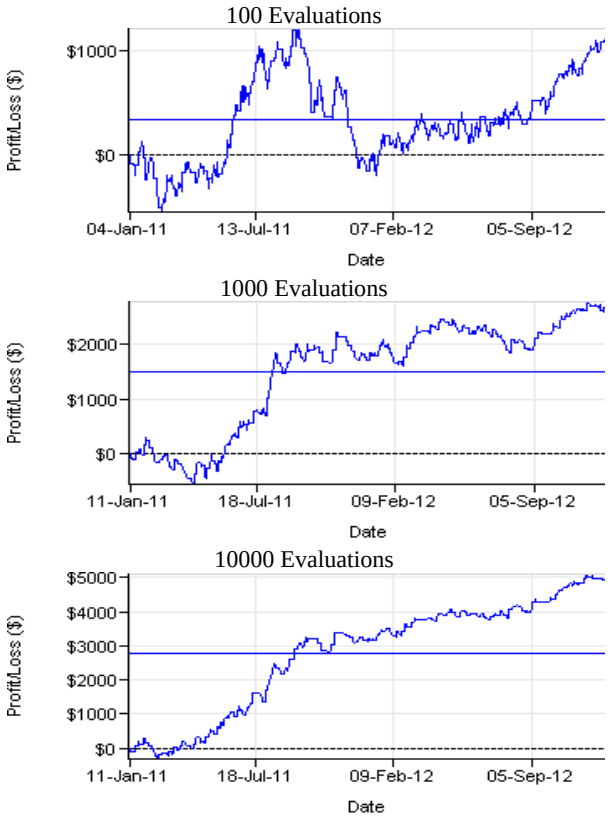

Fig 9. Cumulative profit of the MRA trading system, on the EURUSD instrument, after optimization by the ORCS algorithm for 100, 1000 and 10000 evaluations

The results obtained, from the tests carried on 23 FOREX instruments, by the CS and ORCS algorithms, suggest that both algorithms have been able to present success in the problem of algorithmic trading systems optimization, while ORCS outperformed CS in the majority of tests. These results conform to the results produced from the benchmark functions tested in section 4.

6. DISCUSSIONS AND CONCLUSION

In this paper, the one rank cuckoo search (ORCS) algorithm was proposed, by applying two updates to the behavior of the original cuckoo search (CS) algorithm: first is the one rank (combined evaluation) rule, following a more conservative attitude of consuming the function evaluations, by merging together the new solutions generated from the exploration and exploitation phases to evaluate and rank them jointly, and second is the bound by best solutions rule, to enforce constraints by utilizing the best solutions found so far, while further exploring the search domain. The proposed algorithm has been tested in comparison to the original algorithm, and the experimental results indicate a favorable performance improvement in the greater part of all the benchmark function studied.

The ORCS algorithm uses three parameters only to control its operations. This is considered an advantage, compared to other optimization algorithms like GA and PSO, by reducing the complexity associated with the use of the ORCS algorithm, and allowing researchers to focus their efforts without worrying about the best operational parameters to use, making it more convenient to apply the algorithm to different optimization problems with ease.

The CS and ORCS algorithms have been applied to optimize a real-world problem of algorithmic trading systems optimization with a clear success. The ORCS algorithm outperforms the CS algorithm again, to conform to the experimental results of the benchmark function.

Further research can focus on the study of the algorithm parameters and verify on their sensitivity to affect its performance, studying the usefulness of employing adaptive parameters (ex. an increasing population size), extra study on the behavior of the Lévy flights and the optimal employment of it operations in optimization, the hybridization with other optimization algorithms, applying the algorithm to further real-world problems and evaluating their success, and studying on the automatic detection and prevention of overfitting in the problem of algorithmic trading systems and other optimization problems related to forecasting.

7. REFERENCES

[1] Goldberg, D.E. 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional.

[2] Kennedy, J. and Eberhart, R. 1995. Particle swarm optimization. Neural Networks, 1995. Proceedings., IEEE International Conference on (1995), 1942–1948.

[3] Tan, Y. et al. 2011. Advances in Swarm Intelligence, Part I: Second International Conference, ICSI 2011, Chongqing, China, June 12-15, 2011, Proceedings. Springer.

[4] Baghel, M. et al. 2012. "Survey of Metaheuristic Algorithms for Combinatorial Optimization." International Journal of Computer Applications. 58, 19 (Nov. 2012), 21–31.

[5] Blum, C. and Roli, A. 2003. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison." ACM Computing Surveys (CSUR). 35, 3 (2003), 268–308.

[6] Talbi, E.-G. 2009. Metaheuristics: From Design to Implementation. Wiley.

[7] Yang, X.-S. 2010. Nature-Inspired Metaheuristic Algorithms: Second Edition. Luniver Press.

[8] Yang, X.S. and Deb, S. 2009. Cuckoo search via Lévy flights. Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on (2009), 210–214.

[9] Yang, X.S. and Deb, S. 2010. "Engineering optimisation by cuckoo search." International Journal of Mathematical Modelling and Numerical Optimisation. 1, 4 (2010), 330–343.

[10] Civicioglu, P. and Besdok, E. 2011. "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms." Artificial Intelligence Review. (2011), 1–32.

[11] Chaboud, A. et al. 2009. "Rise of the machines: Algorithmic trading in the foreign exchange market." FRB International Finance Discussion Paper. 980 (2009).

[12] Kissell, R. and Malamut, R. 2006. "Algorithmic decision-making framework." The Journal of Trading. 1, 1 (2006), 12–21.

[13] Miner, R.C. 2008. High Probability Trading Strategies: Entry to Exit Tactics for the Forex, Futures, and Stock Markets. Wiley.

[14] Weissman, R.L. 2004. Mechanical Trading Systems: Pairing Trader Psychology with Technical Analysis. Wiley.

[15] Achelis, S. 2000. Technical Analysis from A to Z, 2nd Edition. McGraw-Hill.

[16] Murphy, J.J. 1999. Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. New York Institute of Finance.

[17] Barthelemy, P. et al. 2008. "A Lévy flight for light." Nature. 453, 7194 (2008), 495–498.

[18] Gutowski, M. 2001. "Lévy flights as an underlying mechanism for global optimization algorithms." arXiv preprint math-ph/0106003. (2001).

[19] Pavlyukevich, I. 2007. "Lévy flights, non-local search and simulated annealing." Journal of Computational Physics. 226, 2 (2007), 1830–1844.

[20] Andrei, N. 2008. "An unconstrained optimization test functions collection." Adv. Model. Optim. 10, 1 (2008), 147–161.

[21] Molga, M. and Smutnicki, C. 2005. "Test functions for optimization needs." Test functions for optimization needs. (2005).

[22] Bussieck, M.R. and Pruessner, A. 2003. "Mixed-integer nonlinear programming." SIAG/OPT Newsletter: Views & News. 14, 1 (2003), 19–22.

[23] Levinson, M. 2009. Guide to Financial Markets. Bloomberg Press.

[24] Sharpe, W.F. 1994. "The Sharpe Ratio." The Journal of Portfolio Management. 21, 1 (Jan. 1994), 49–58.

[25] Everitt, B.S. and Skrondal, A. 2010. The Cambridge Dictionary of Statistics. Cambridge University Press.

[26] Hawkins, D.M. 2004. "The problem of overfitting." Journal of chemical information and computer sciences. 44, 1 (2004), 1–12.

[27] Lin, L. et al. 2005. "Genetic algorithms for robust optimization in financial applications." Computational Intelligence. 2005, (2005), 387–391.