Optimisation of Differential Evolution for Automated Parameter-Response Trading Agents

Finn Formica
Department of Computer Science
University of Bristol
Bristol BS8 1UB, U.K.
qu19756@bristol.ac.uk

Abstract—**Differential Evolution (DE) is a powerful algorithm that can be applied to any general problem to determine the global maximum of the problem. To improve the performance of adaptive automated trader agents within financial markets the DE algorithm DE/rand/1 was applied to a parameter-response zero-intelligence trader - PRDE - and yielded increase in profit and economic efficiency. This paper optimises the control parameters of the DE/rand/1 algorithm as well as proposes a new trader agent - PRJ - using the JADE algorithm. These were both implemented and tested utilising the Bristol Stock Exchange as well as rigorous hypothesis testing to ensure confidence in the results. PRJ was shown to adapt to changing market conditions faster than PRDE, and generate increased profit across a static market, dynamic market, and supply shock with perfectly elastic supply and demand ranges.**

Index Terms—**Automated Trading, Financial Markets, Market Simulation, Adaptive Trader-Agents, Bayesian optimisation, Differential Evolution, JADE.**

I. Introduction

Most financial markets today are home to a large number of automated trading agents that are trying to maximise profits. These trading agents, which started out as simple algorithms following predetermined strategies, have now evolved to adapt their strategies to changing market conditions. Automated trading agents have several advantages over human traders, including the ability to execute orders at faster speeds and with higher frequencies, and the ability to make more rational and efficient trades by reducing the impact of human emotions [1], [2]. However, these agents can be difficult to model and understand, particularly in complex and nonlinear environments such as financial markets [3], [4].

The Bristol Stock Exchange (BSE) is a simplified simulation of financial markets that allows for a deeper understanding of how these markets function, while still allowing for complex behaviours to emerge when populated with adaptive automated trading agents. One such agent, developed by Cliff and known as PRZI-Differential-Evolution (PRDE) [5], uses a technique called Differential Evolution (DE) to converge on an optimal trading strategy. DE evaluates a set of vectors and iteratively improves them through a process of selection, crossover, and mutation, ultimately converging on the globally optimal solution. DE is particularly useful in nonlinear and complex systems because it can produce optimal agents without the need for an accurate model of the system.

PRDE, which uses the DE/rand/1 algorithm, showed a doubling of economic efficiency when compared to another adaptive algorithm called PRSH - PRZI with Stochastic Hill-climbing. However, the DE control parameters were not optimised for performance. This paper aims to evaluate the performance of PRDE as the DE parameters k and F are varied in order to optimise the trader's profit per second. Additionally, the paper will examine the use of a self-adaptive DE scheme called JADE, which aims to address some of the criticisms of DE/rand/1, such as the trial-and-error process of optimising control parameters and the poor convergence performance compared to other adaptive DE algorithms. A new trader is proposed, PRZI-JADE (PRJ - pronounced 'purge'), using the JADE algorithm to provide self-adaptive control parameters and reduce the trial-and-error required to optimise the trading agent. PRJ's performance is compared to PRDE's in various market conditions, including static and dynamic markets and supply shocks simulating black swan events like the 2008 Financial Crash. To ensure confidence in results, BSE simulations are run multiple times and statistical tests are applied to the data.

The contributions of the paper are summarised as follows:

- Optimisation of DE control parameters for PRDE to maximise profit per second.
- Proposal of a new self-adaptive algorithm, PRJ, that adapts DE control parameters to the market conditions and requires less hyper-parameter tuning.
- Evaluation of PRJ against PRDE across a range of market dynamics using BSE.

II. Background

In this section, the background of the report is presented. The Bristol Stock Exchange is discussed alongside explanations of how the traders on the simulation behave, the idea of Differential Evolution is introduced and some of its extensions which improve on the original algorithm, and finally several methods for the hyper-parameter tuning of the adaptive traders are evaluated.

A. Bristol Stock Exchange

The exchange simulation Cliff used was the Bristol Stock Exchange, an open-source modern financial exchange operating on a Continuous Double Auction (CDA) matching bids and asks via a central Limit Order Book (LOB). The LOB stores a record of each unmatched bid and ask order submitted to the exchange from the traders, and completes any transaction where there is a cross in the spread. To simulate the continuous nature of a real financial exchange, each 'second' of time within the simulation is discretised into equal steps, allowing every trader to interact with the exchange once a second.

BSE can be populated with any number of automated trading strategies available in the codebase. Some traders are zero-intelligence, issuing their bids or asks via some pre-determined non-adaptive algorithm, and others are capable of adapting to the market conditions through Machine Learning. The trading agents available on BSE are as follows [5–8]:

- Zero-Intelligence Constrained (ZIC): A zero-intelligence algorithm that randomly generates quote prices that do not generate a loss.
- Shaver (SHVR): A zero-intelligence algorithm that adds or subtracts a penny from the best bid or ask in order to have the best quote on the LOB.
- Giveaway (GVWY): A zero-intelligence algorithm that quotes its limit price.
- Parametrised-Response Zero Intelligence (PRZI): A parameterised zero-intelligence algorithm that is controlled by a parameter $s \in [-1, 1]$ which makes it behave like either GVWY or SHVR at either extreme and ZIC when $s = 0$.
- PRZI-Stochastic Hill-climber (PRSH): An adaptive algorithm that varies the parameter s of PRZI using a stochastic hill-climber algorithm to determine the most optimal value of s for the market conditions.
- PRZI-Differential Evolution (PRDE): An adaptive algorithm like PRSH but using differential evolution to find the optimal value of s.

Once the market is populated, the traders are supplied with limit orders within a pre-determined supply and demand range. These ranges can be set to model static and dynamic markets as well as supply and demand shocks. BSE traders can only buy and sell in a single unit. If a buyer purchases a unit for less than its limit price, the difference is the profit it makes - the same is true if a seller sells a unit for more than its limit price. Since the absolute value of profit can drastically vary depending on the time spent trading, the traders value of profit is divided by time to produce the profit per second metric (pps). The adaptive trader strategies are evaluated on their pps.

B. Differential Evolution

Differential evolution (DE) is a population-based optimisation algorithm introduced by Storn and Price that operates over a continuous vector space [9]. DE is particularly well-suited for optimising problems that are expensive to evaluate or have complex constraints. It is a global optimisation method, meaning that it is not limited to finding local minima/maxima, but rather can find the global minima/maxima of a function.

For a population size of k, vectors $x_i, i = 1, 2, \cdots, k$ are initially chosen at random with a uniform probability distribution to represent the continuous space. DE generates a new vector by adding the weighted difference between two vectors to a random third vector - this operation is known as mutation. To increase the diversity of the perturbed vector, crossover is introduced, which mixes the parameters of the mutated vector with those of a pre-determined vector. The mutated vector is then evaluated against a member of the population and the vector with the maximum utility is retained - this is known as selection. The processes of mutation, crossover, and selection are iteratively applied to the population allowing the DE algorithm to effectively search the space of possible vectors and converge on an optimal solution. For PRDE, since the evaluated vector space $s \in [-1, 1]$ contains only one dimension, there is no need for crossover within the DE algorithm, and the vectors are only mutated and then selected.

The different DE schemes come in the form DE/x/y.

- x represents which vector the weighted difference is applied to for mutation.
- y is the number of difference vectors used.

The above scheme is known as DE/rand/1 since a random vector is chosen each time for mutation. Each mutant vector is generated according to:

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}) \tag{1}$$

where $F \in [0, 2]$ is the differential weight, and $r_1, r_2, r_3 \in \{1, 2, \cdots, k\}$ are mutually different random indexes within the population and not equal to the current index i being evaluated. To permit these conditions $k \geq 4$.

Other popular DE schemes utilised in the literature are:

- DE/best/1

$$v_i = x_{best} + F \cdot (x_{r_1} - x_{r_2}) \tag{2}$$

- DE/current-to-best/1

$$v_i = x_i + F \cdot (x_{best} - x_i) + F \cdot (x_{r_1} - x_{r_2}) \tag{3}$$

where x_{best} is the best performing vector in the population. These schemes improve on DE/rand/1, providing faster convergence by including information about the best performing vector in the population. Although these algorithms have been shown to improve the performance of DE, hyper-parameter tuning is required to ensure that the optimal parameters are used [10] and the value of F is fixed. Instead, Zhang *et al.* [11] implemented the JADE algorithm which is self-adapting; updating the values for the differential weight and crossover probability iteratively as the vector performances are evaluated.

The JADE algorithm is a variant of DE that uses an archive of previously unsuccessful strategies. It generates a mutation vector v_i by combining the current vector x_i, a randomly selected vector from the top $100p\%$ performing vectors in the population x^p_{best}, and a randomly chosen vector from the union of the current population P and the archive A as follows:

$$v_i = x_i + F \cdot (x^p_{best} - x_i) + F \cdot (x_{r_1} - \tilde{x}_{r_2}) \tag{4}$$

where $p \in [0,1]$ and $\tilde{x}_{r_2}$ is randomly chosen from $P \cup A$. The archive is a collection of vectors that have failed selection and is used to improve overall diversity and provide information about progress direction. It is maintained at a size of k by randomly removing some solutions at the end of each generation.

The differential weight parameter F is updated using a Cauchy distribution with location parameter μ_F and scale parameter 0.1. If $F > 1$, it is truncated to 1, and if $F \leq 0$, it is regenerated. The location parameter μ_F is initialised at 0.5 and is updated each generation as follows:

$$\mu_F = (1-c) \cdot \mu_F + c \cdot \text{mean}_L(S_F) \tag{5}$$

$$F = \text{randc}(\mu_F, 0.1) \tag{6}$$

where $c \in [0,1]$ is a positive constant, S_F is the set of all successful differential weight parameters in the generation, and the Lehmer mean $\text{mean}_L(S_F)$ is calculated as:

$$\text{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{7}$$

Traditional DE has two control parameters than need to be tuned for optimal performance, the differential weight and crossover parameter. These parameters are known to be specific to the situation and so require some form of optimisation to find appropriate values. JADE also has two control parameters, c and p which control the rate of parameter adaption and the greediness of the mutation strategy respectively. However, Zhang *et al.* [11] shows how these parameters are insensitive to different problems mitigating the issue of choosing optimal parameters. JADE performs best with $\frac{1}{c} \in [5, 20]$ and $p \in [5\%, 20\%]$ where the life span of μ_F ranges from 5 to 20 generations and the top $5-20\%$ strategies are considered in the mutation process.

C. Hyper-parameter tuning

Hyper-parameter tuning is the process of selecting optimal sets of hyper-parameters for a machine learning algorithm. These hyper-parameters are settings that control the learning behaviour of the algorithm. The most common method for hyper-parameter tuning is Grid Search, which involves discretising the hyper-parameters uniformly across the parameter space, and evaluating the model for each combination of parameters. This method is exhaustive and can be very time-consuming, especially as the number of hyper-parameters increases or the discretisation step decreases [12].

To address this issue, Random Grid Search was developed which randomly samples the hyper-parameter values from a random distribution rather than the fixed grid. This can be more efficient than Grid Search, especially if the search space is large and the optimal values are not evenly distributed. However, both of these algorithms are 'naive' since they do not take into account previous attempts of optimisation to infer future tests of hyper-parameter values [12].

Instead, Bayesian optimisation is a more sophisticated approach that uses Bayesian inference to model the distribution of the performance of the model as a function of the hyper-parameters. It then uses this model to select the next set of hyper-parameters to try, with the goal of finding the optimal set as quickly as possible. Bayesian optimisation has been shown to be particularly effective for hyper-parameter tuning, especially for models with a large number of hyper-parameters or with a particularly expensive cost function [13].

In the case of BSE, running the simulations for long enough to converge is extremely time expensive and so using Grid Search or Random Grid Search may take several days of continuous computing. For this reason Bayesian optimisation was utilised to tune the hyper-parameters of PRDE.

III. Exploring the Hyper-Parameters of PRDE

In this section, the key parameters of the DE/rand/1 algorithm within PRDE were evaluated and optimised. Cliff intentionally used the minimum viable population size $k = 4$ and kept the DE coefficient at the standard value $F = 0.8$ for simplicity. These parameters were used as a benchmark of performance.

A. Experiment Design

PRDE was tested across two different market supply and demand curves - a perfectly elastic supply and demand which replicated the experiment Cliff conducted when evaluating PRDE against PRSH, and a unitary elastic supply and demand which is a more realistic model of a financial market. For the perfectly elastic simulation the buyers were instructed to buy for no more than 200 and the sellers were instructed to sell for no less than 100; and for the unitary elastic simulation the supply and demand curves ranged from $(100, 200)$ each. In each simulation 20 PRDE trader agents with identical parameters were used. The traders were fed orders randomly within a 10 second interval utilising the time-mode 'drip-jitter'. According to Cliff [5], it took PRDE approximately 25 days of continuous trading to show convergence of the profit per second (pps), so the simulations were run for 30 days to ensure convergence across all trials. The Python package `bayes_opt` was used to implement Bayesian optimisation [14] and run the `market_session()` function within the BSE. The average pps across all traders in the simulation was returned to the wrapper after each trial to evaluate and infer future parameter values through iterative trials, with 80 trials in total, 15% of which were dedicated to random exploration and the rest for exploitation of the Bayesian inference.

Once a set of parameters were determined for each supply and demand setup, a second experiment was run to validate the results. This involved running two BSE simulations for each supply and demand setup. The first simulation used PRDE agents with the parameters $(k, F) = (4, 0.8)$, while the second validation experiment used PRDE agents with the parameters determined from the Bayesian optimisation. The results from these experiments were averaged to produce two sets of data, which were then tested for statistical significance.

To account for the stochastic nature of the evolution process, each simulation contained 20 trials configured as described above.

B. Experiment Results

Running on a M1 MacBook Pro each Bayesian optimisation took roughly 22 hours each. Neither the perfectly elastic simulation nor unitary elastic simulation showed convergence of the profit per second of the trading agents during the optimisation process which is unusual considering the high number of iterations the algorithm was run for. However, this outcome is likely due to the highly random nature of the evolution process of PRDE which could cause large fluctuations within the results.

The profit per second was plotted against F, with the colour of the data point varied according to the value of k. Fig. 1, the results for the unitary elastic simulation, shows that the highest profit per second was achieved with parameters $(k, F) = (9, 1.766)$ to yield a maximum of pps $= 1.096$.

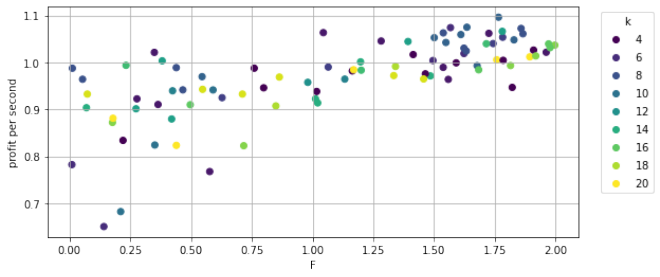

Fig. 1. pps against k and F for unitary elastic supply and demand curves

Fig. 2, the results for the perfectly elastic simulation, shows that the highest profit per second was achieved with parameters $(k, F) = (8, 0.9953)$ to yield a maximum of pps $= 3.849$. It seemed that a higher mutation coefficient was favoured in the unitary elastic simulation whereas a mid-range mutation coefficient was favoured in the perfectly elastic simulation.

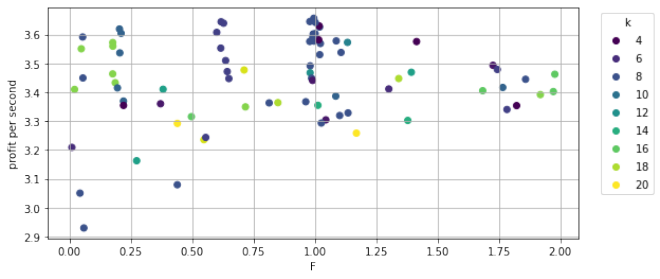

Fig. 2. pps against k and F for perfectly elastic supply and demand curves

The second experiment utilised these parameters for the trader agents in the market simulation across 20 trials - each experiment of 20 trials took 17 hours on a M1 MacBook Pro. Fig. 3 and Fig. 4 display the averaged results for the unitary elastic and perfectly elastic simulations respectively. For the unitary elastic simulation the optimised parameters were shown to have a clear advantage over those taken from Cliff [5]. The optimised parameters took much longer to converge than the original parameters, which is likely due to the increased population size. The original parameters had a mean pps of 0.9557 whereas the optimised parameters had a mean pps of 1.106 which is an increase of 15.7%.

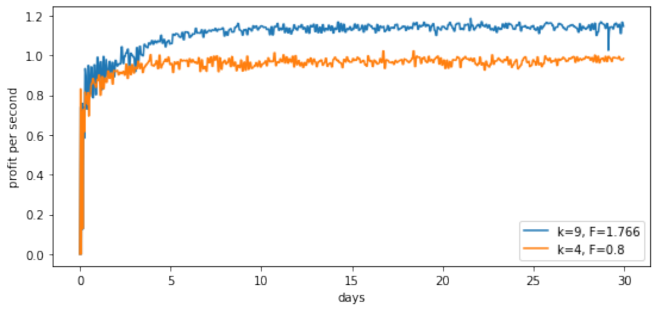

Fig. 3. Profit per second of PRDE in unitary elastic simulation

For the perfectly elastic simulation the optimised parameters also had an improved performance with the original parameters yielding a mean pps of 3.806 and the optimised parameters a mean pps of 3.925 for an increase in pps of 3.13%.

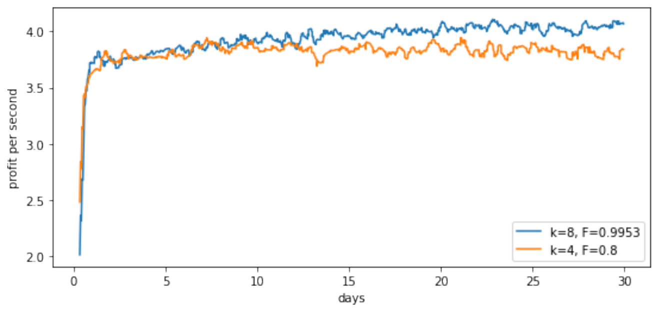

Fig. 4. Profit per second of PRDE in perfectly elastic simulation

To ensure confidence in results statistical testing was completed. To determine whether the data was normal the Shapiro-Wilk statistical test was used. For a significance level of $\alpha = 0.05$, the test statistics and p-values for the Shapiro-Wilk test are shown in Table I. All p-values were less than 0.05 and so the null hypothesis that the data sets were normal was rejected with 95% confidence. The non-parametric Wilcoxon test was utilised to determine the statistical significance of the difference in means.

TABLE I
MEAN PPS AND SHAPIRO-WILK TEST STATISTICS AND P-VALUES FOR PRDE

	Unitary elastic		**Perfectly elastic**	
trader	$k = 4$	$k = 9$	$k = 4$	$k = 8$
mean pps	0.9557	1.106	3.806	3.925
statistic	0.3754	0.4776	0.3843	0.4506
p-value	4.484×10^{-4}	1.783×10^{-4}	0.0	0.0

The null hypothesis for the Wilcoxon test was no significant difference between the mean pps of the original and optimised parameters for each market condition. With a significance level of $\alpha = 0.05$ the test statistics and p-values are shown in Table II. For both sets of results the p-value was less than the significance level, and hence the null hypothesis was rejected with 95% confidence and it was concluded that the optimised parameters from the Bayesian optimisation were superior when compared with those used by Cliff [5].

TABLE II
WILCOXON TEST STATISTICS AND P-VALUES FOR PRDE

	Unitary elastic	**Perfectly elastic**
statistic	645.0	57550
p-value	2.288×10^{-118}	7.726×10^{-69}

IV. Extending PRDE with JADE

In this section, the research on PRDE was extended by replacing the DE/rand/1 algorithm with JADE, which utilised DE/current-to-$p$best/1 with optional archive. PRJ was tested against PRDE across a variety of market scenarios and the performance of each trader type was evaluated.

A. Experiment Design

The following experiments were run utilising two types of trader agents - PRDE with Cliff's control parameters and PRJ. Each experiment contained 20 trader agents each of the same type and with the same parameters. The trader agents were tested across three different market scenarios - static market, dynamic market, and a supply shock. The experiments were conducted across 50 days of continuous trading to allow the traders more time to adapt during changing market conditions. Furthermore, traders were fed orders randomly within a 10 second interval utilising the time-mode 'drip-jitter'. Due to limited time and compute resources the market scenarios were only simulated with perfectly elastic supply and demand curves.

The static market had the same perfectly elastic supply and demand ranges as previously with buyers paying no more than 200 and sellers selling for no less than 100; the dynamic market was a model of a ranging market where the supply and demand ranges started the same as the static market but were slowly changed over time using a sinusoidal offset function with an amplitude of 50 and added Gaussian noise; and the supply shock was a model of an asset becoming less scarce and hence the price of it rapidly decreases and so the supply range started at $(200, 200)$ and dropped to $(100, 100)$ halfway through the simulation with the demand range fixed at $(250, 250)$ for the entire duration.

The control parameters for PRJ were chosen to maintain the lifespan of μ_F over 10 generations and select the top 20% of successful strategies for the mutation process, $c = 0.1$ and $p = 0.2$. Finally, the population size of $k = 15$ was chosen so that the 3 best strategies could be randomly picked during the mutation process.

B. Experiment Results

Running on a M1 MacBook Pro each set of 20 trials took roughly 17 hours when testing PRJ and 26 hours when testing PRDE. Fig. 5 shows the averaged results for the dynamic market simulation. PRJ (blue) seemed to be able to adapt much more quickly to the changing market conditions and capture extra profit as the supply and demand ranges varied. PRDE (orange) appeared to be affected negligibly by the varying ranges. Overall PRJ had a mean pps of 4.094 and PRDE 3.783 which is an increase of 8.21%.

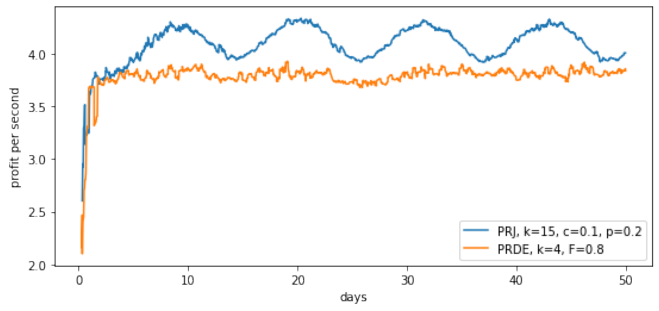

Fig. 5. Profit per second of PRDE and PRJ in a dynamic market

Fig. 6 shows the averaged results for the supply shock simulation. PRJ initially converged to a higher pps than PRDE before the shock, and then managed to increase its pps dramatically during the event before settling down to a similar steady-state value as PRDE. Again, this shows PRJ was able to adapt to the changing market conditions much quicker than PRDE and increase its economic efficiency. Overall PRJ had a mean pps of 4.591 and PRDE 4.471 which is an increase of 2.69%.

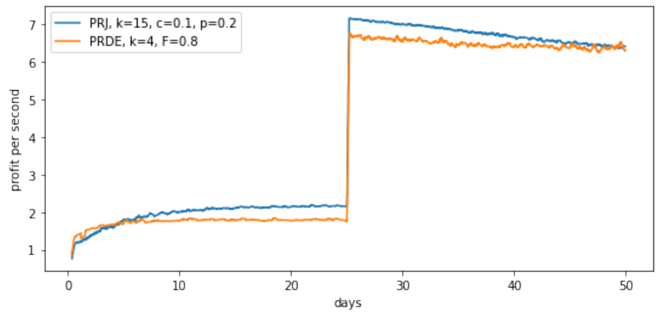

Fig. 6. Profit per second of PRDE and PRJ in a supply shock

Fig. 7 shows the averaged results for the static market simulation. In this simulation PRJ took longer to converge, however, achieved a higher pps overall once it did. Overall PRJ had a mean pps of 4.010 and PRDE 3.763 which is an increase of 6.56%. The pps percent gain produced by PRJ is double that produced by the optimised PRDE in the perfectly elastic static market simulation.

To ensure confidence in results statistical testing was completed. To determine whether the data was normal the Shapiro-Wilk statistical test was used. For a significance level of

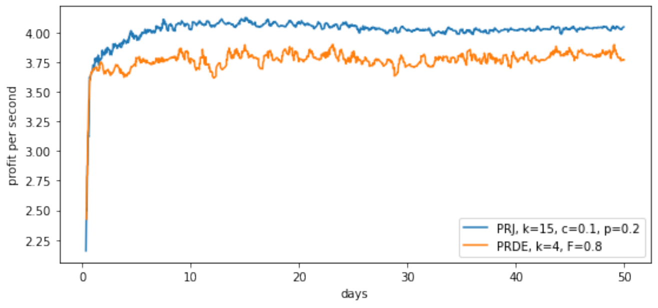

Fig. 7. Profit per second of PRDE and PRJ in a static market

$\alpha = 0.05$, the test statistics and p-values for the Shapiro-Wilk test are shown in Table III. All p-values were equal to 0.0 and so the null hypothesis that the data sets were normal was rejected with 95% confidence. The non-parametric Wilcoxon test was utilised to determine the statistical significance of the difference in means between PRJ and PRDE.

TABLE III
MEAN PPS AND SHAPIRO-WILK TEST STATISTICS AND P-VALUES FOR PRJ AND PRDE

	Static		**Dynamic**		**Shock**	
trader	PRJ	PRDE	PRJ	PRDE	PRJ	PRDE
mean pps	4.010	3.763	4.094	3.783	4.591	4.471
statistic	0.2710	0.4147	0.5734	0.2884	0.7279	0.6900
p-value	0.0	0.0	0.0	0.0	0.0	0.0

The results of the Wilcoxon tests are shown in Table IV. For a significance level of $\alpha = 0.05$ all three results were below the significance level, which meant that the null hypothesis was rejected with 95% confidence. Therefore, it was concluded that PRJ on average made more profit that PRDE in all three market conditions with perfectly elastic supply and demand ranges. This demonstrates the superiority of JADE against DE/rand/1.

TABLE IV
WILCOXON TEST STATISTICS AND P-VALUES FOR PRJ AND PRDE

	Static	**Dynamic**	**Shock**
statistic	17310	17280	214000
p-value	5.083×10^{-217}	4.874×10^{-221}	2.025×10^{-113}

V. DISCUSSION AND FUTURE WORKS

This paper optimised the DE trader agent presented by Cliff [5]. This was accomplished initially through hyper-parameter tuning of the DE/rand/1 parameters k and F utilising Bayesian optimisation for perfectly elastic and unitary elastic supply and demand ranges. This hyper-parameter tuning method was successful in increasing the profit per second of the PRDE trader agent, showing statistically significant gain across both market simulations.

The proposed trader agent PRJ, utilising the JADE algorithm, was also tested against PRDE across three market scenarios - static, dynamic, and supply shock - with perfectly elastic supply and demand ranges. PRJ was shown to outperform PRDE in all three market scenarios and adapted faster to the market condition in the dynamic market and supply shock. These results were achieved through the simulation of a real market environment utilising the Bristol Stock Exchange and confidence in the results was provided by rigorous hypothesis testing.

Despite the dominance of PRJ against PRDE in the perfectly elastic simulations, these supply and demand ranges are not particularly accurate to real financial markets, and further testing with unitary elastic ranges would be beneficial. Furthermore, PRJ should be tested against more risk-based events like positive and negative supply and demand shocks to determine its robustness in the event of a market crash or other such events that can often cause the loss of huge amounts of money. Finally, all sets of experiments were previously run homogeneously, i.e. each trial contained the same trader type with the same parameters. To observe how PRJ behaves when faced with a multitude of algorithms it should be tested in a market with a combination of Zero-Intelligent traders like ZIC and SHVR as well as some adaptive traders like PRDE and ZIP to simulate trading in a real financial market more accurately.

REFERENCES

[1] I. Yagi, Y. Masuda, and T. Mizuta, “Analysis of the impact of high-frequency trading on artificial market liquidity,” *IEEE Transactions on Computational Social Systems*, vol. 7, no. 6, pp. 1324–1334, 2020.
[2] D. MacKenzie, “Trading at the speed of light: How ultrafast algorithms are transforming financial markets,” 2021.
[3] G. Nuti, M. Mirghaemi, P. Treleaven, and C. Yingsaeree, “Algorithmic trading,” *Computer*, vol. 44, no. 11, pp. 61–69, 2011.
[4] D. Sornette and D. Sornette, “Why stock markets crash: Critical events in complex financial systems,” 2017.
[5] D. Cliff, “Metapopulation differential co-evolution of trading strategies in a model financial market,” 2022.
[6] ——, “Parametrised-response of zero-intelligence traders,” https://arxiv.org/abs/2103.11341.
[7] D. K. Gode and S. Sunder, “Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality,” *Journal of political economy*, vol. 101, no. 1, pp. 119–137, 1993.
[8] J. Rust, J. H. Miller, and R. Palmer, “Behavior of trading automata in a computerized double auction market,” in *The Double Auction Market Institutions, Theories, and Evidence*. Routledge, 2018, pp. 155–198.
[9] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.
[10] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, “A comparative study of differential evolution variants for global optimization,” in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, pp. 485–492.
[11] J. Zhang and A. C. Sanderson, “Jade: Adaptive differential evolution with optional external archive,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.
[12] T. Yu and H. Zhu, “Hyper-parameter optimization: A review of algorithms and applications,” *CoRR*, vol. abs/2003.05689, 2020. [Online]. Available: https://arxiv.org/abs/2003.05689
[13] V. Nguyen, “Bayesian optimization for accelerating hyper-parameter tuning,” in *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 2019, pp. 302–305.
[14] F. Nogueira, “Bayesian Optimization: Open source constrained global optimization tool for Python,” ”https://github.com/fmfn/BayesianOptimization”, 2014–.