Comparison of Adaptive Differential Evolution (ADE) and Genetic Algorithm (GA) for Optimizing Trading Strategies

---

Abstract:

In this report, we compare the performance of the Adaptive Differential Evolution (ADE) algorithm and the Genetic Algorithm (GA) for optimizing trading strategies. Both algorithms are applied to a trading problem that involves finding the optimal decision variables, which include the RSI window size, RSI_buy, and RSI_sell thresholds, to maximize the final amount of cash in a trading account. The performance of both algorithms is analyzed, and their strengths and weaknesses are discussed.

1. Introduction:

Trading strategies often involve the use of technical indicators, such as the Relative Strength Index (RSI), to identify buy and sell signals. Optimizing the parameters of these indicators can lead to better trading performance. In this study, we apply two evolutionary algorithms, ADE and GA, to optimize the RSI-based trading strategy and compare their performance.

2. Problem Formulation:

The trading strategy optimization problem involves maximizing the final amount of cash in a trading account by adjusting three decision variables: RSI window size, RSI_buy, and RSI_sell thresholds. The RSI_buy and RSI_sell thresholds determine the buy and sell signals, while the RSI window size is used to calculate the RSI values.

3. Adaptive Differential Evolution (ADE):

The Adaptive Differential Evolution (ADE) algorithm is an extension of the Differential Evolution (DE) algorithm, which incorporates adaptive control parameters to improve the search efficiency. The ADE algorithm is applied to the trading strategy optimization problem, which involves finding the optimal RSI window size, RSI_buy, and RSI_sell thresholds to maximize the final amount of cash in the trading account. The ADE algorithm consists of four main steps: DE/rand/1 mutation, crossover, TLLS-based local search, and crowding-based selection.

3.1 DE/rand/1 Mutation:

In the DE/rand/1 mutation, a mutated vector (vi) is created for each individual in the population. The mutation process involves selecting three random individuals (r1, r2, r3) from the population, with the constraint that they are distinct from each other and from the target individual (i). The mutated vector is then calculated using the following formula:

vi = r1 + F * (r2 – r3)

where F is the scale factor for mutation.

3.2 Crossover:

The crossover operation combines the target individual (xi) and the mutated vector (vi) to produce a trial vector (ui). For each dimension (j) in the problem, the trial vector's components are determined by comparing a random number with the crossover rate (CR). The trial vector's component (ui,j) is set to the corresponding component of the mutated vector (vi,j) if the random number is less than CR, otherwise, it is set to the corresponding component of the target individual (xi,j).

3.3 TLLS–based Local Search:

The TLLS–based local search improves the exploitation ability of the proposed ADE by encouraging individuals to explore their surrounding regions, thereby increasing solution accuracy. In this approach, TLLS is adopted as the local search strategy and is applied to every individual in the population, unlike the original TLLS, which is applied with a probability. The pseudo–code of the TLLS–based local search can be found in lines 5–8 of Algorithm 1. First, for each individual in the current population, two offspring (v1 and v2) are generated using a Gaussian distribution centered around the trial vector (ui) with a dynamically adjusted standard deviation (sigma). The Gaussian distribution is represented as:

v_i = Gaussian(x, sigma)

The standard deviation (sigma) is adjusted using the following formula:

sigma = 10^(–1 – (10 / (D + 3)) * (gen * n + i) / MaxFEs)

where D is the number of dimensions, gen is the current generation, n is the population size, i is the individual index, and MaxFEs is the maximum number of function evaluations.

The offspring individual (offspring_population[i]) is chosen as the best among the trial vector (ui) and the two random vectors (v1, v2) based on their fitness (the final amount of cash in the trading account). This approach ensures that individuals explore their surrounding regions, improving the exploitation ability of the ADE algorithm and increasing the solution accuracy.

3.4 Crowding–based Selection:

After the global reproduction and TLLS–based local search, the crowding–based selection is carried out on the 2n offspring to obtain the population in the next generation. This selection strategy ensures diversity preservation.

In the crowding–based selection process, each offspring finds its nearest neighbor in the current population. If the offspring is better than its nearest neighbor, the nearest neighbor is replaced by the offspring; otherwise, the offspring is discarded.

Let u_i^t represent the offspring and x_nn^t represent its nearest neighbor. The mathematical format of crowding–based selection for the maximization of MMOP is shown as:
x_nn^{t+1} = u_i^t, if fit(u_i^t) > fit(x_nn^t)
x_nn^{t+1} = x_nn^t, otherwise
This crowding–based selection strategy ensures that only the best offspring are included in the next generation population while maintaining diversity within the population. As a result, the algorithm converges to the optimal solution more efficiently and effectively.

After iterating through the specified number of generations (MaxFEs // n), the best individual in the final population is identified, and its corresponding fitness value and decision variables are returned. This individual represents the optimal solution to the trading strategy optimization problem.

4. ## Genetic Algorithm (GA):

5. ## Comparison of ADE and GA:

6. ## Conclusion: