LLM-Enabled Automated Algorithm Design for Multiuser Fluid Antenna Communications

Abstract—**Fluid antenna is a new reconfigurable antenna technology that can dynamically adjust the positions or ports of radiating elements and therefore provides a new degree of freedom for wireless communications. However, the associated port selection is a challenging large-scale combinatorial optimization problem and difficult to solve. Existing manually designed heuristic algorithms are not only labor-intensive, but cannot achieve satisfactory performance. In this paper, we propose a novel paradigm that leverages large language models (LLMs) for automated design of optimization algorithms for fluid antenna systems without manual hyperheuristic tuning. Specifically, we study the problem of maximizing the minimum signal-to-interference-plus-noise ratio (SINR) in the downlink to ensure fairness among users by optimizing port selection and beamforming. We investigate two LLM-enabled algorithm optimization strategies. The first is to optimize the crossover and mutation operations to enhance the performance of the well-known genetic algorithm and the second is to design AutoPort, a new heuristic from scratch by LLM, to solve the optimization problem. Simulation results verify that the proposed method can achieve near-optimal performance and significant improvement over the conventional genetic algorithm and the deep learning approach.**

Index Terms—**Fluid antenna, port selection large language models, evolution of heuristic, automated heuristic design**

I. Introduction

Multiple-input multiple-output (MIMO) technology has been a cornerstone for enhancing performance of wireless communications over several decades. However, the gains of MIMO come at a cost. Take massive MIMO in the fifth-generation (5G) mobile communications as an example. The deployment of massive MIMO antenna arrays entails significant capital expenditures, including the cost of numerous antennas, space, radio frequency (RF) chains and energy consumptions. It is therefore desirable to develop new cost-effective antenna solutions that can achieve performance comparable to massive MIMO, but without incurring its deployment and operational costs.

Recently, a novel reconfigurable antenna architecture known as the fluid antenna system (FAS) has been proposed. FAS broadly encompasses any software-controlled fluidic, dielectric, or conductive structures, such as liquid-based antennas [1], pixel-based antennas [2], and metasurfaces [3] that are capable of dynamically reconfiguring their position, orientation, and other radiation characteristics. This reconfigurability enables the realization of spatial diversity within a physically constrained environment. The concept of FAS was first introduced to the wireless communications domain by Wong et al. in [4]. Since then, it has attracted significant interests to develop efficient FAS for the sixth-generation (6G) mobile communications systems. The fundamental concept of FAS involves the dense deployment of a large number of antenna ports within a limited physical space, while utilizing only a single RF chain [5] and therefore substantially reduces the costs. The diversity analysis has revealed tremendous gain (more than 10x) of FAS-enabled receivers [6] and the diversity-multiplexing tradeoff has been characterized from an information-theoretic perspective in [7].

FAS has been exploited together with the MIMO technology to boost the performance of both single-user and multiuser systems by leveraging conventional antenna arrays and flexible port positions. However, MIMO-FAS also introduces new optimization challenges. The port selection is a high-dimensional combinatorial problem and is coupled with the traditional beamforming optimization, resulting in a difficult large-scale and highly non-convex problem. There have been various port selection methods for FAS [8], [9]. When both the MIMO transmitter and receiver have fluid antennas, quantum computation was studied in [10] and reduced exhaustive search and alternating optimization based on joint convex relaxation were proposed in [11] to maximize the channel capacity. For the optimization of multiuser MIMO-FAS, alternating optimization techniques that iteratively optimizes port position and beamforming while keeping the other fixed, is often employed [12] [13]. However, these methods not only incur high complexity when solving subproblems at each iteration, but also cannot guarantee satisfactory performance.

Deep learning has emerged as a new tool for optimization of MIMO-FAS. Deep reinforcement learning has been applied to port selection and beamforming with promising performance gains [14], [15]. Graph neural networks (GNNs) have been studied which adopt a two-stage framework to solve the two subproblems of optimizing port position and beamforming separately [16], but this decoupling may ignore the relation between the two subproblems. More recently, large pre-trained large language models (LLMs) have been introduced to address the mobility issue in MIMO-FAS [17]. It utilizes LLMs as a pre-trained neural network together with low-rank adaptation as a fine-tuning method to predict future channel state information (CSI) from historical CSI, and then select ports with higher accuracy in medium-to-high mobility scenarios. Intelligent FAS empowered by LLM has been proposed in [18] where LLM has been applied for channel extrapolation, channel prediction, precoder design and autonomous FAS cooperation. Nevertheless, deep learning based methods still face the challenges of large training data requirement, high training complexity, tedious parameter tuning, poor generalization and limited explainability for MIMO-FAS.

Recently, there has been a new paradigm that leverages LLM to automatically design heuristic algorithms for various applications [19]. This approach utilizes an LLM's ability to comprehend algorithms and generate code to iteratively create heuristics, significantly reducing human design efforts. Unlike black-box deep-learning methods, the resulting heuristics are interpretable. For example, evolution of heuristics (EoH) [20], [21] uses an evolutionary framework to refine a population of heuristics with an LLM. Each heuristic is defined by its conceptual idea and code implementation. The LLM generates new or improved heuristics, which are then evaluated on target problem instances. The population is subsequently updated by retaining the best-performing heuristics. This cycle of generation, evaluation, and selection drives the automated discovery of effective heuristic algorithms. Similarly, Funsearch [22] adopts LLMs in an evolutionary search framework for automated function search for mathematical discoveries. Instead of using a simple population management method, a multi-island approach is adopted to enhance diversity. This paradigm has been extended and applied to diverse optimization algorithm design tasks, including routing [21], scheduling [23], integer linear programming [24], and Bayesian optimization [25].

This paper proposes an LLM-enabled framework for the optimization of multi-user FAS downlink multiple-input-single-output (MISO) communications systems, with the objective to maximize the minimum signal-to-interference-plus-noise ratio (SINR) across users under the total transmit power constraint. The optimization problem entails port selection and the corresponding beamforming design. While given the port selection, an efficient algorithm to optimize the beamforming is available in the literature [26], the optimization of port selection is a challenging problem. Different from existing methods in the literature that manually design algorithms to optimize port selection and beamforming directly, our proposed method searches for heuristic algorithms automatically to solve the problem. In other words, we optimize algorithms instead of variables. Our key contributions are summarized as follows:

1) This paper presents the first LLM-enabled algorithm design framework for SINR maximization in multi-user FAS downlink multiple-input-single-output (MISO) communications systems. This framework addresses limitations of existing methods, which automates the creation of customized and fully interpretable heuristic algorithms without domain knowledge, pre-trained data or parameter tuning. It not only minimizes manual efforts, but also leads to superior performance and potentially brings insights on new algorithms design.
2) We propose two design strategies to solve the port selection problem. The first is based on the genetic algorithm (GA), and we utilize LLM to optimize the crossover and mutation functions. The second is Auto-Port, completely designed by prompting LLMs, which adopts an evolutionary search framework, iteratively searching for algorithms that lead to higher SINR.
3) Extensive simulations demonstrate the effectiveness of our proposed method. Results show that the first strategy can significantly improve the performance over the baseline GA while the second strategy can achieve even better and near-optimal user performance.

The remainder of this paper is organized as follows. Section II introduces the multiuser FAS downlink system model and details of the optimization problem. Section III presents the preliminaries of LLM-enabled automated algorithm design. Section IV introduces details of the two design strategies to enhance the GA and design a new heuristic algorithm. Section V provides numerical evaluations and performance comparisons. Finally, concluding remarks and discussions on future directions are given in Section VI.

Notation: Bold lowercase and uppercase letters denote vectors and matrices, respectively. Superscripts $(\cdot)^T$ and $(\cdot)^\dagger$ represent the transpose and Hermitian transpose. $\mathbb{C}^{m\times n}$ and $\mathbb{R}^{m\times n}$ denote the sets of $m \times n$ complex- and real-valued matrices, respectively. The notation $\mathcal{CN}(\boldsymbol{\mu}, \boldsymbol{\Gamma})$ denotes a circularly symmetric complex Gaussian distribution with the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Gamma}$. $\|\mathbf{a}\|$ denotes the Frobenius norm of a vector $\mathbf{a}$. $|\cdot|$ denotes the absolute value of a scalar or the cardinality of a set.

II. System Model and Problem Formulation

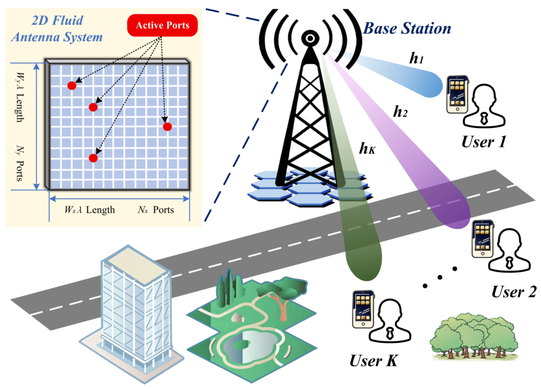

Fig. 1. System model of a MISO downlink communications system using 2D-FAS at the BS.

We consider a multiuser MISO-FAS downlink system as illustrated in Fig. 1, in which the base station (BS) equipped with a two-dimensional (2D) FAS array serves K users each with a single fixed antenna. The 2D FAS array has a dimension of $W = \lambda W_x \times \lambda W_y$, where λ denotes the carrier wavelength. λW_x and λW_y are the array lengths in the x and y directions. The array consists of $N = N_x \times N_y$ ports uniformly distributed across the aperture, with N_i ports aligned along each dimension λW_i for $i \in \{x, y\}$. We assume the BS has n RF chains and therefore only select or activate n out of N ports to serve K users and it is required that $n \geq K$.

To characterize the correlation between any two ports in the 2D array, we assume a rich scattering environment and employ the classical Jake's model for the correlation [27]. Specifically, the spatial correlation between the (n_x, n_y)-th and the (n_i, n_j)-th ports is expressed as

$$\begin{aligned} J_{(n_x,n_y),(n_i,n_j)} &= J_0\left(\frac{2\pi d}{\lambda}\right), \\ &= J_0\left(2\pi\left(\frac{|n_x-n_i|}{N_x-1}W_x\right)^2+\left(\frac{|n_y-n_j|}{N_y-1}W_y\right)^2\right) \end{aligned} \tag{1}$$

where $d=\sqrt{\left(\frac{|n_x-n_i|}{N_x-1}\lambda W_x\right)^2+\left(\frac{|n_y-n_j|}{N_y-1}\lambda W_y\right)^2}$ denotes the distance between these two ports, and J_0 is the zero-order spherical Bessel function.

For notation convenience, we map the 2D port coordinate to the 1D port index n_{n_x,n_y} as follows:

$$n_{n_x,n_y}=(N_y-1)n_y+n_x\in\{1,\cdots,N\}. \tag{2}$$

We can therefore construct the spatial correlation matrix $\mathbf{J}\in\mathbb{R}^{N\times N}$ of the 1D channel between any two ports using (1) and (2).

Apparently the matrix $\mathbf{J}$ is symmetric, and its eigenvalue decomposition is

$$\mathbf{J}=\mathbf{U}\Lambda\mathbf{U}^\dagger, \tag{3}$$

where $\mathbf{U}$ contains the eigenvectors and Λ is a diagonal matrix and its diagonal elements are eigenvalues.

Then the channel from all antenna ports at the BS to user k can be expressed as

$$\mathbf{h}_k=\sqrt{\rho_k}\mathbf{U}\sqrt{\Lambda}\boldsymbol{g}_k, \tag{4}$$

where ρ_k denotes the large-scale path loss, $\boldsymbol{g}_k\in\mathbb{C}^{N\times1}$ denotes the small-scale fading channel vector that follows the complex Gaussian distribution, i.e., $\mathcal{CN}(\mathbf{0},\boldsymbol{I})$.

Suppose the set of chosen ports is denoted as ϕ which contains n non-repeated ports, and the resulting channel of user k is $\mathbf{h}_k(\phi)\in\mathbb{C}^{n\times1}$. Denote the transmit signal and the beamforming vector for user k as s_k with unit power and $\mathbf{w}_k$, respectively. The received signal at user k is

$$z_k=\mathbf{h}_k^T(\phi)\mathbf{w}_k s_k+\sum_{j=1,j\neq k}^{K}\mathbf{h}_k^T(\phi)\mathbf{w}_j s_j+n_k, \tag{5}$$

where n_k is the additive white Gaussian noise with zero mean and variance σ^2. The received SINR of user k is

$$\gamma_k=\frac{|\mathbf{h}_k^T(\phi)\mathbf{w}_k|^2}{\sum_{j=1,j\neq k}^{K}|\mathbf{h}_k^T(\phi)\mathbf{w}_j|^2+\sigma^2}. \tag{6}$$

The design objective is to maximize the minimum SINR of all users subject to the total power constraint P by optimizing the port selection and the beamforming vectors. Mathematically, the problem is formulated as

$$\max_{\{\mathbf{w}_k\},\phi} \quad \gamma \tag{7}$$

$$\text{s.t.} \quad \gamma_k\geq\gamma,\forall k, \tag{8}$$

$$\phi\subset\{1,\cdots,N\},|\phi|=n, \tag{9}$$

$$\sum_{k=1}^{K}\|\mathbf{w}_k\|^2\leq P. \tag{10}$$

Given ϕ or equivalent channels $\{\mathbf{h}_k(\phi)\}$, the problem (7) has been well studied and the solution to find the optimal $\{\mathbf{w}_k\}$ is available based on the uplink-downlink duality [26]. Therefore in this paper we focus on the optimization of port selection ϕ.

Assume given ϕ and the corresponding beamforming vectors, the optimal SINR is $\gamma(\phi)$ by using the algorithm in [26]. Then the problem (7) can be simplified and rewritten as

$$\begin{aligned} \max_{\phi} \quad & \gamma(\phi) \\ \text{s.t.} \quad & \phi\subset\{1,\cdots,N\},|\phi|=n. \end{aligned} \tag{11}$$

The above selection problem is to choose n out of N ports to maximize the SINR. If we use the exhaustive search to solve it, the number of all possible combinations is $\binom{N}{n}$, and its complexity grows even faster than the exponential increase. When n or N is large, the number of possible solutions is huge and cannot be enumerated. For instance, when $N=64$, the numbers of all possibilities for $n=4$ and $n=8$ are 635,376 and 4.4262×10^9, respectively. In this paper, we will leverage LLMs to design efficient heuristic algorithms to solve the problem (11).

III. Preliminaries: LLM-enabled Automated Algorithm Design

Recently, a new paradigm, often referred to as LLM-driven automated heuristic design (AHD), has emerged that leverages LLMs to automate the design of interpretable heuristic algorithms, significantly reducing the reliance on tedious hand-crafting and domain model training. At its core, this paradigm treats the heuristic itself as the target of an optimization search. Instead of searching for a solution to a problem instance, it searches the space of possible heuristics to find one algorithm that performs well across many problem instances. To formalize this, we can define the AHD process as follows.

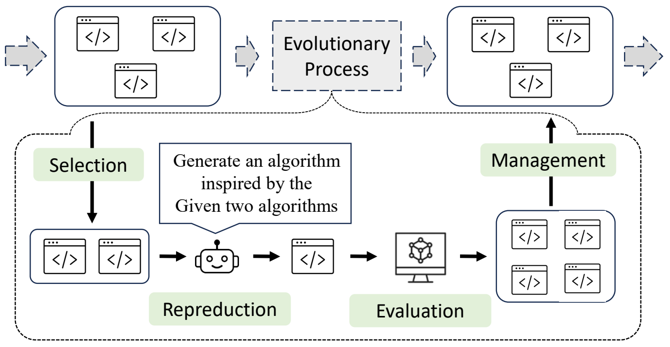

Fig. 2. The pipeline of LLM-enhanced automated heuristic design.

Let the heuristic design task be denoted by T. For our paper, this task is to design a heuristic for the port selection problem defined in (11). We evaluate any candidate heuristic on a set of given problem instances, I, where each instance $i\in I$ corresponds to a specific channel setting. The goal is to find an optimal heuristic h^* from a vast space of possible heuristics, H^S, that maximizes a performance metric, $F(h,I)$. In our context, this metric is the average minimum SINR achieved by a heuristic h across all channel instances in I. This can be expressed as:

$$F(h, I) = \frac{1}{|I|} \sum_{i \in I} f(h, i), \quad h \in H^S, \tag{12}$$

where $f(h, i)$ is the performance score (i.e., the resulting minimum SINR $\gamma(\phi)$) obtained by applying heuristic h to instance i. The optimal heuristic h^* is the one that solves the following optimization problem:

$$h^* = \arg\max_{h \in H^S} F(h, I). \tag{13}$$

Since the algorithm space H^S is immense and unstructured, solving (13) directly is intractable. AHD addresses this challenge by employing an evolutionary search framework powered by LLM [21]. Fig. 2 illustrates the general pipeline used by LLM-enabled AHD. In this framework, each heuristic h is directly represented as a code implementation (e.g., Python implementation) and/or its natural language description [21]. The search begins with an initial "population" of algorithms and iteratively improves them over generations. The process is summarized as follows:

Step 1: Initialization An initial population of candidate algorithms, $P_0 = \{h_1, h_2, \ldots, h_n\}$, is created. This is done by providing the LLM with a description of the optimization task T and asking it to generate a diverse set of initial strategies.

Step 2: Iterative Process To evolve the population from the t-th generation P_t to $t+1$-th population P_{t+1}, the following steps are repeated:

- **Step 2.1: Selection** "Parent" algorithms h_p are selected from the current population P_t according to certain criteria (e.g., with probability proportional to their fitness values).
- **Step 2.2: Reproduction** New "offspring" algorithms h_o are created from "parent" algorithms h_p. It is guided by specific instructions (prompts) given to the LLM. For example, the LLM might be prompted to "combine the strategies of these two parent algorithms" or "suggest a modification to this algorithm to improve its performance". This can be abstractly represented as:

  $$h_o = \text{LLM}(\text{Prompt}(h_p)).$$

  Here, `Prompt()` constructs the instructions and provides the parent algorithm(s) as context, and `LLM()` represents the LLM's inference process that generates the new offspring algorithm.

  An example prompt used for reproduction in the bin packing problem [29] is illustrated in the following box:

  Example Prompt for Reproduction

  I'm designing a heuristic for the bin packing problem. The goal is to minimize the number of used bins. At each step, an item is placed into the bin with the maximum score.

  Here are two existing heuristics:

  No.1 Heuristic description:

  `<Code for heuristic 1>`

  No.2 Heuristic description:

  `<Code for heuristic 2>`

  Design a new heuristic inspired by, but different from, the ones provided.

  1. Identify the common idea behind the provided heuristics.
  2. Describe your new heuristic in one sentence.
  3. Implement it in a Python function `score(item, bins)` that returns the `scores`.

- **Step 2.3: Evaluation** The fitness $F(h_o, I)$ of each new algorithm h_o is measured. This is a practical step where the algorithm's code is executed on the set of channel instances I. The average performance score, as defined in (12), determines the algorithm's fitness.
- **Step 2.4: Management** A new population P_{t+1} is formed by selecting the fittest algorithms from the combined pool of the current population P_t and the new offspring h_o. This survival-of-the-fittest mechanism ensures that effective algorithmic ideas are carried forward and refined in subsequent generations.

Step 3: Termination The evolutionary search loop continues until a stopping criterion is met, such as a maximum number of generations or a predefined budget for LLM queries. The final output is the best-performing algorithm $\hat{h}^*$ discovered during the search.

The LLM-driven AHD paradigm offers the following distinct advantages over existing methods:

- Unlike manual design, it automates the discovery process, reducing human effort and potentially uncovering novel, non-intuitive strategies.
- Compared to deep learning methods, which produce "black-box" neural networks, the final output of AHD is a fully interpretable, human-readable algorithm (e.g., a Python function). This transparency allows for verification and analysis, enabling researchers to gain new insights into the problem structure. Furthermore, AHD does not require the extensive data collection and lengthy training time associated with deep learning models, and generalizes well to unseen scenarios.

In this paper, our research utilizes the LLM4AD platform [28], an open-source Python library we have recently developed to advance research in LLM-driven algorithm design. LLM4AD features a modular architecture that decouples high-level search strategies from the underlying language models and task-specific evaluations. This platform includes modules for search methods, a unified LLM programme template, and a sandboxed environment for securely evaluating candidate algorithms. LLM4AD is easy to use, and we only need to specify the latter two modules for a specific problem, i.e., the programme template and the evaluation function. It supports a wide range of applications with an extensive library of around 100 tasks across optimization, machine learning, and scientific discovery.

IV. Proposed Optimization Strategies

In this section, we present two strategies to use LLM for designing efficient algorithms to solve the problem (11). The first one optimizes parts of operations of a basic GA and thus enhance its performance, while the second one prompts the LLM to design a complete algorithm by itself. For implementation, we employ EoH available in the LLM4AD platform.

A. Optimization of GA

We first describe the principle of a basic GA, and then introduce LLM to optimize two components of its operations: crossover and mutation.

1) Basic GA: The GA is a heuristic algorithm widely used to solve constrained and unconstrained optimization problem that is inspired by the natural selection.

It iteratively refines a population of individual solutions. Briefly speaking, at each iteration, the GA selects individuals from the current population to be parents and uses them to produce the children or offsprings via operations including crossover and mutation for the next iteration. This process is repeated until convergence, e.g., for a fixed number of iterations or until no further improvement is made, and hopefully the population will evolve to an optimal solution.

In this paper, the specific basic GA employed to solve (11) is summarized as follows.

- Initialization. The algorithm begins with random initial population with M (an even number) individual port selection solutions. Each solution contains n non-repeated ports.
- Create new populations of solutions by repeating the following steps at each iteration until convergence.
  - Population Selection. It evaluates the M individual port selection solutions by using the objective function in (11), and then select the best half $\frac{M}{2}$ individuals as parents. It also reserves a small portion, e.g., $Mp, (0 < p < 1)$ individuals of the best solutions, called elite individuals, that will be passed to the next iteration directly without any further processing.
  - Crossover. It generates $M - Mp$ children solutions from $\frac{M}{2}$ parents solutions. The basic crossover function is to split ports in the parents solutions in half (assuming n is an even number), randomly choose $M - Mp$ solutions from each half solutions (which could be repeated), and then combine them together to form $M - Mp$ new solutions.
  - Mutation. It introduces random changes in the population after crossover. This helps maintain diversity in the population and explores new solution space. The basic mutation function used is to uniformly sample each port selection in the population and then randomly change it to a different port.
  - Validation. After the above operations, the obtained individual port selection solutions may contain repeated ports and are therefore invalid. This step will identify these solutions and replace repeated ports with randomly chosen non-repeated ones.
  - Population merging. This step will merge the above obtained population with the elite individuals to form a new population of size M to be used for the next iteration.
  - Evaluation. In this step, the performance of the population will be evaluated using the objective function in (11) and the best will be stored for the current iteration.
- The best solution across iterations will be retained as the solution to the problem (11).

2) Optimizing Crossover: It can be seen from the above description that in the basic GA, although the general purposes of crossover and mutation are clear, their implementations can vary according to the problem definitions and there is no optimal implementation in general. This gives rise to the optimization of the specific implementation for crossover and mutation functions. In this subsection, we first keep the basic mutation function and all other parts of the basic GA fixed, and only optimize the crossover function using EoH provided in the LLM4AD platform.

Specifically, for EoH to automate the process of creating optimized crossover function, we provide EoH the programme template in Python as shown in Listing 1. In the template, we need to include details of the input and output parameters together with a clear and concise task description. Then EoH will instruct the LLM to evolve the heuristic design using the template, together with the task evaluate function based on the objective function of (11) and channel data.

```
template_program = '''
def crossover(parents, M):
    """
    Design a crossover function to be used in
        the genetic algorithm.
    Args:
     parents: 2D integer Numpy array of shape
         (n_parents, n).
     M: Number of output populations.
    Returns:
    offspring: 2D integer Numpy array of shape
        (M, n).
    """
'''
task_description = "Design a crossover
    function for a genetic algorithm. The
    function performs a genetic crossover
    function on parents to generate multiple
    offspring."
```

Listing 1. The programme template for the crossover function.

EoH will then iteratively refine the population of the crossover function by using the LLM until the termination condition is met. When evaluating a candidate crossover function, the LLM4AD will run the basic GA algorithm introduced in the previous subsection with the candidate crossover function.

By checking the log of the algorithm evolution, we have found some typical sample crossover algorithms designed by EoH as shown in Table I. We can see that the evolution has explored a broad range of possible crossover algorithms. The finally chosen crossover function that gives the best

TABLE I
TYPICAL CROSSOVER ALGORITHMS ATTEMPTED IN THE EVOLUTION.

No.	Description
1	The algorithm performs a uniform crossover with probability-based gene selection, where each gene in the offspring is independently chosen from either parent with equal probability, creating highly diverse combinations while maintaining genetic material from both parents.
2	The algorithm performs a dynamic segment swap crossover by randomly varying segment lengths and using a probabilistic swap threshold to create offspring, enhancing exploration while maintaining genetic diversity.
3	The algorithm performs a circular shift crossover by rotating parent arrays at random pivot points and combining segments from the rotated parents to create offspring, introducing rotational diversity in recombination.
Chosen	The algorithm performs a gene frequency-based crossover with probabilistic inheritance, where each gene in the offspring is selected from either parent based on the relative frequency of that gene's value across the entire parent population.

TABLE II
TYPICAL MUTATION ALGORITHMS ATTEMPTED IN THE EVOLUTION.

No.	Description
1	The mutation function randomly selects a subset of genes in each individual and replaces them with new random values within the allowed range to ensure exploration. ***Comment***: This is the start of the evolution and is very much like the basic algorithm.
2	The mutation function applies a random cyclic shift to each row of the population array, preserving all elements but rearranging their positions.
3	The new algorithm introduces diversity by flipping a variable number of bits in each individual, where the number of bits flipped is randomly chosen between 1 and 20% of the individual's length, and the mutation values are drawn from a Gaussian distribution centered around the original values with a standard deviation of $\frac{N}{4}$.
Chosen	The new algorithm introduces diversity by swapping pairs of elements in 50% of individuals with 70% probability, then applying uniform random noise to 15% of elements in each individual, scaled by $\frac{N}{5}$ and rounded to integers.

performance for the training data is also shown in Table I and the Python code is given in Listing 4 of Appendix A. We can see that the optimized crossover function first calculates the frequency-based probabilistic inheritance from parents solutions, and then each port selection in the offspring is selected from two randomly chosen parents based on the relative frequency of that port's value across the entire parent population.

3) Optimizing Mutation: To further enhance the performance of GA, next we optimize the mutation function to explore the solution space while using the above optimized crossover function. The programme template in Python for LLM to automate the evolution of the mutation function is provided in Listing 2 below. Similar to the crossover function, the template describes the input and output parameters, as well as the task.

```
template_program = '''
def mutation(population, N):
    """
    Design a mutation function to be used in
        the genetic algorithm.
    Args:
        population: Numpy array of shape (p, n).
        N: The largest integer for the
            population is N-1.
    Returns:
        mutated_population: Numpy array of
            shape (p, n).
    """
'''
task_description = "Design a mutation
    function for a genetic algorithm. The
    function modifies a given 2D population
    array parents to ensure exploration of
    the genetic algorithm."
```

Listing 2. The programme template for the mutation function.

By checking the log of the algorithm evolution, we have found some typical sample mutation algorithms designed by EoH as shown in Table II. Again we can see that the evolution has explored a broad range of possible mutation algorithms.

The finally chosen mutation function that gives the best performance for the training data is also shown in Table II and the Python code is given in Listing 5 of Appendix B. As can be seen, rather than simply modifying the port selection in a uniform way, the optimized mutation function is sophisticated. Specifically, it introduces diversity by swapping pairs of elements in 50% of individual solutions with 70% probability, and then applies random noise that follows the uniform distribution $[-\frac{N}{5}, \frac{N}{5}]$ to 15% of port selections in each individual solution and finally rounds them to integers. We have observed that the number of mutated solutions, the probabilities and the distribution with the associated parameters have all been adjusted during the algorithm evolution.

As will become clear later from simulation results, with both optimized crossover and mutation functions, the updated GA shows significant performance gain over the basic GA.

B. AutoPort: Direct Search of A Heuristic Algorithm

Different from the above strategy that only modifies part of an algorithm, here we introduce the second strategy called EoH that utilizes LLM to design a complete heuristic algorithm without any prior restriction on the family of algorithms that can be used for port selection. To this end, we need to provide EoH more detailed instructions about the port selection task, which also involves solving the SINR balancing problem given a port selection and this function is given as sinr_balancing().

The programme template is given in Listing 3 which provides the descriptions of the input and output parameters, and the specific task. It guides the LLM to search for an effective heuristic algorithm that solves the port selection problem for a block of B channel realizations.

```
template_program = '''
def select_ports(K,n,N,Pt,B,H,noise_power):
    """
   Select n out of N ports to maximize the
       average objective value of B
       communications channels.

    Args:
       K: The number of users.
       n: The number of ports to be selected.
       N: Total number of ports available.
       Pt: Total transmit power available.
       B: Total number of channel realizations.
       H: Numpy array of shape (B,N,K). It
           denotes B channel realizations.
       noise_power: Noise power.
    Returns:
       port_sample: Numpy array of shape (B,
           n). For each row of it, all values
           should be integers from 0 to N-1
           and cannot be repeated.

    For the n-th channel realization H(n),
        suppose port is a valid port selection
        solution, and then the effective
        channel becomes h_n = H[n,port,:]. The
        objective value will be calculated
        using the pre-defined function
        f_n=sinr_balancing(n, K, h_n, Pt,
        noise_power).
    """
'''
task_description = "Implement a function that
    selects a subset of ports for each
    channel realization to maximize the
    average SINR of B communications
    channels."
```

Listing 3. The programme template for the AutoPort function.

TABLE III
TYPICAL AUTOPORT ALGORITHMS ATTEMPTED IN THE EVOLUTION.

No.	Description
1	The new algorithm uses a greedy approach with local search, where ports are selected iteratively by adding the port that maximizes the immediate gain in SINR, followed by a local optimization step to swap ports for further improvement.
2	The new algorithm uses a simulated annealing approach with adaptive cooling schedule, where port selection is optimized through temperature-controlled probabilistic acceptance of new solutions, focusing on both exploration and exploitation.
3	The new algorithm uses particle swarm optimization with adaptive velocity updates and neighborhood topologies to efficiently explore the port selection space while maintaining diversity and convergence.
Chosen	The new algorithm uses a randomized beam search approach, where multiple candidate subsets are generated and refined in parallel, then the best subset is selected based on the objective value.

Some typical algorithms in the evolution and the finally chosen algorithm are shown in Table III. The code for the chosen AutoPort function in Python is depicted in Listing 6 of Appendix C. We can see that a number of heuristic methods including greedy algorithm, adaptive simulated annealing and particle swarm optimization have been attempted during the evolution.

The finally chosen AutoPort algorithm uses a greedy randomized adaptive search procedure (GRASP) [30]. The GRASP methodology was developed in the late 1980s for solving hard optimization problems. GRASP is a multi-start metaheuristic algorithm consisting of two phases: a constructive randomized adaptive phase and a local search phase. In the first phase, the algorithm constructs multiple candidate solutions by randomly selecting ports with probabilities weighted by channel gains of all ports. Instead of choosing the best out of those greedy solutions, the GRASP algorithm improves each greedy solution in the second phase by applying a local search method to explore the local solution space. In the local search, the algorithm modifies one selected port of a greedy solution at a time, and evaluates the modified solution to check if it is better. The best of the local search solutions is returned as the final solution. Applying local search starting from different greedy solutions may lead to a variety of different locally optimal solutions, and the best of which will be returned as the output solution.

V. Simulation Results and Discussions

In this section, numerical experiments are conducted to evaluate and validate the proposed LLM-enabled optimization for port selection in a fluid antenna downlink MISO system.

A. Simulation Settings

The simulation setup follows the description in Section II. Specifically, the system is operating with the carrier frequency $f_c = 2$ GHz, with the BS transmit power of 20 dBm, the noise power spectral density of -174 dBm/Hz, and the bandwidth of 10 MHz. Unless otherwise specified, we consider the BS is equipped with a 2D FAS with the port configuration $N_x = N_y = 8$ and dimension $W_x = W_y = 2$. The number of activated ports is the same as the number of users, i.e., $n = K$.

We assume the distance between the BS and users is $d_k = 200$ m, $\forall k$, and the large-scale path loss is given by

$$\rho_k = 128.1 + 37.6 \log_{10}(d_k/1000). \tag{14}$$

For each setting, we generate 1,000 channel data for LLM-enabled algorithm optimization and 100 data for the evaluation of all schemes.

We use Deepseek-v3 as the default LLM. For the basic GA and optimized GA, 100 and 50 iterations are used. The population size is 20 and 20% of it are chosen as elite solutions. We use 50 channel realizations to optimize algorithms using EoH and 100 channel realizations to evaluate the performance. Note that all LLM designed algorithms are optimized only once with the above settings and $K = 8$. The optimized algorithms are then generalized to other unseen settings during the evaluation.

B. Evaluation and Comparison Schemes

We evaluate the performance of the following LLM-enabled schemes:

- GA-C. This is the GA with EoH-optimized crossover operation.

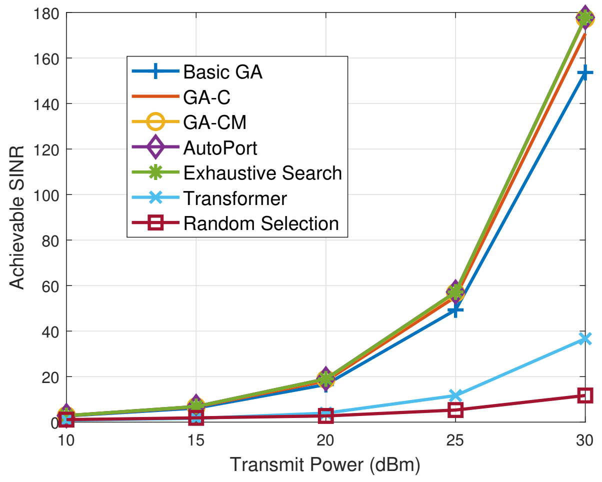

Fig. 3. Achievable SINR vs the transmit power for $K = 4$.

- GA-CM. This is the GA with EoH-optimized crossover and mutation operations.
- AutoPort. This is the heuristic algorithm completely optimized by EoH.

We also compare them with the following benchmark schemes:

- Exhaustive Search. It provides the optimal performance but is only possible when $\binom{N}{n}$ is not too large.
- Basic GA. This is the GA with basic crossover and mutation operations.
- Transformer based solution [31]: As a representative method of deep learning, we adopt a transformer with an eight-head attention module. It requires 10,000 training data.
- Random Selection. This scheme randomly chooses n out of N antenna ports.

C. Performance Comparison

We first consider a relatively small system which serves $K = 4$ users. The achievable SINR is depicted in Fig. 3 as the transmit power varies. We can observe that all three LLM-optimized algorithms achieve close to optimal performance while the performance of the transformer and the random search is unsatisfactory. The transformer also exhibits limited performance that is even worse than the basic GA because it cannot effectively extract features for the high-dimensional port selection problem. To better illustrate the performance difference, we also show the results in Table IV that is normalized by the optimal solution using Exhaustive Search. We can clearly see that GA-C has significantly improved the performance of the basic GA by optimizing the crossover operation. By further optimizing the mutation operation, GA-CM achieves more than 99% of the optimal performance. It is interesting to observe that AutoPort, the scheme that is directly designed by EoH, is even better than GA-CM, and can achieve the optimal performance.

To further test the generalization ability of the proposed method, we extend the testing scenario where the small

TABLE IV
NORMALIZED SINR BY THE EXHAUSTIVE SEARCH VS THE TRANSMIT POWER, $K = 4$.

P (dBm)	10	15	20	25	30
Basic GA	0.9458	0.8978	0.8678	0.8623	0.8640
GA-C	0.9769	0.9736	0.9512	0.9622	0.9602
GA-CM	0.9975	0.9940	0.9918	0.9941	0.9975
AutoPort	1.0000	1.0000	1.0000	1.0000	1.0000
Transformer	0.3143	0.2387	0.2080	0.2042	0.2062
Random	0.3843	0.2744	0.1433	0.0929	0.0659

TABLE V
NORMALIZED SINR BY THE EXHAUSTIVE SEARCH VS THE TRANSMIT POWER, $K = 4$, RICIAN CHANNEL.

P (dBm)	10	15	20	25	30
Basic GA	0.9794	0.9662	0.9490	0.9426	0.9237
GA-C	0.9901	0.9825	0.9634	0.9654	0.9597
GA-CM	0.9997	0.9990	0.9995	0.9975	0.9970
AutoPort	0.9999	1.0000	1.0000	1.0000	1.0000
Transformer	0.7738	0.6859	0.7196	0.7909	0.8375
Random	0.3248	0.2348	0.1362	0.0703	0.0316

scale fading follows the Rician distribution with the Rician factor of 5 dB (note that the default setting follows the Rayleigh distribution). The results are given in Table V. We can see that similar to the results in Table IV, all proposed LLM-enabled approaches demonstrate excellent generalization performance, and AutoPort still achieves optimal or near-optimal performance. The performance of the transformer and random schemes is still unsatisfactory.

In the rest of this section, we consider a system with $K = 8$ users. Because of the excessively large number of parameters needed by the transformer, it is excluded for the comparison. We first investigate the convergence performance of difference methods as shown in Fig. 4. For AutoPort, apart from Deepseek-V3, we have also tested other LLMs including Deepseek-R1, a family of open reasoning models with performance approaching that of leading models, Claude 3.7 Sonnet, the first hybrid reasoning model developed by Anthropic and Gemma3, one of the most capable models that runs on a single GPU. We can observe that all LLMs for AutoPort converge fast (within 100 function samples) to similar SINRs, although Deepseek-R1 shows slower convergence. The GA-C and GA-CM methods converge even faster than AutoPort because they only optimize part of the GA.

Next we show the achievable SINR vs the transmit power in Fig. 5. It is observed that as the transmit power increases, LLM-optimized algorithms greatly outperforms the basic GA and the random selection schemes. We normalize the achievable SINR of schemes by that of the basic GA and present the results in Table VI. A closer look at the results reveals that AutoPort achieves slightly better performance than GA-CM and nearly 1 dB gain over the basic GA scheme when the transmit power is 30 dBm.

We then investigate the impact of the number of ports on the achievable SINR with a fixed area of the 2D antenna panel, and results are shown in Fig. 6. As the number of ports increases, it is expected that the performance of all schemes will improve

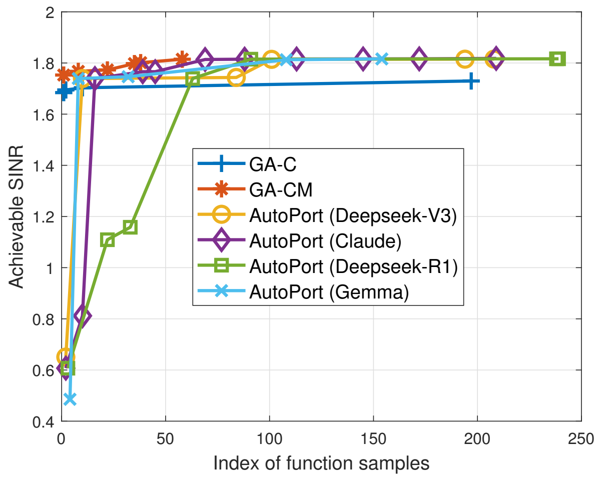

Fig. 4. Convergence behaviour for $K = 8$.

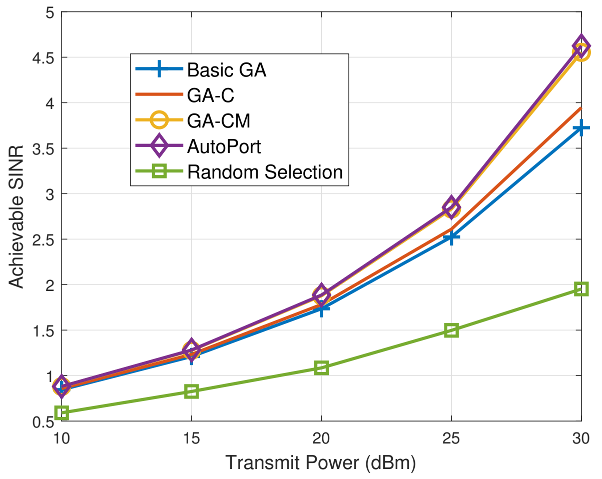

Fig. 5. Achievable SINR vs the transmit power for $K = 8$.

because of the increased spatial degrees of freedom. However, we can see that the achievable SINR of the random, basic GA and GA-C schemes all degrade with increasing number of ports. This can be explained by the fact that a higher number of ports, on the one hand increases the correlation between ports, and on the other hand increases the complexity of port selection, and therefore the resulting SINR of these three schemes becomes worse. By contrast, the GA-CM and AutoPort can achieve more stable performance as the number of ports increases and maintains an increasing performance trend, which verifies their effectiveness.

Finally, we evaluate the impact of size of the 2D antenna array W_x or W_y on the achievable SINR and results are given in Fig. 7. It is seen that the achievable SINR improves as the array size increases. This is expected as large array size reduces the correlation between ports and can better exploit the diversity gain. GA-C only achieves modest performance gain over the basic GA while with further optimized mutation, GA-CM outperforms the basic GA by more than 20% when

TABLE VI
NORMALIZED SINR BY THE BASIC GA VS THE TRANSMIT POWER, $K = 8$.

P (dBm)	10	15	20	25	30
GA-C	1.0157	1.0196	1.0269	1.0340	1.0594
GA-CM	1.0413	1.0555	1.0822	1.1216	1.2224
AutoPort	1.0425	1.0590	1.0870	1.1291	1.2416
Random	0.6991	0.6823	0.6254	0.5931	0.5241

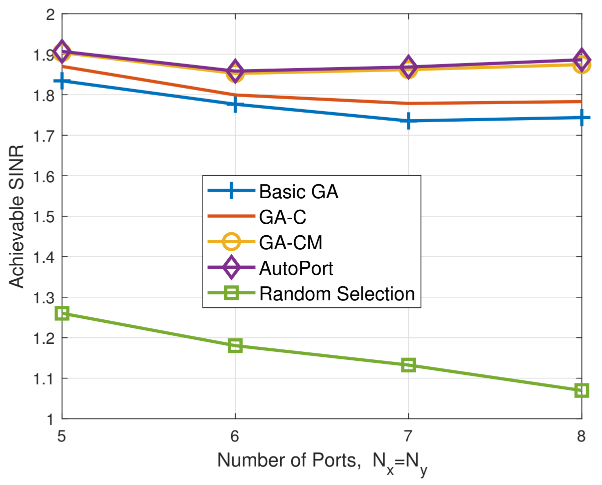

Fig. 6. Achievable SINR vs the number of ports N_x or N_y for $K = 8$.

$W_x = 3$ and approaches the performance of the AutoPort scheme. These results again shown the advantages of the GA-CM and the AutoPort schemes.

VI. CONCLUSIONS

This paper presents an LLM-enabled automated algorithm design method for port selection in multiuser fluid-MIMO systems. Instead of directly optimizing the port selection, this method designs heuristic algorithms automatically to optimize

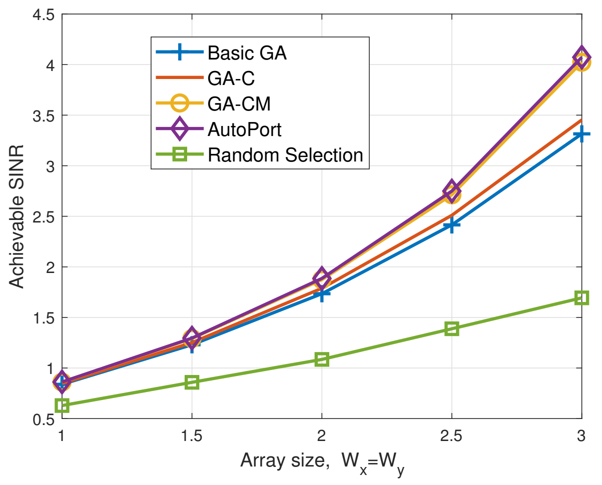

Fig. 7. Achievable SINR vs the size of the 2D antenna array W_x or W_y for $K = 8$.

the port selection using LLM. We have introduced two strategies of this method. One is to optimize the crossover and mutation operations in the GA while the other directly designs the complete heuristic algorithms. We have applied both strategies to the problem of maximizing the minimum SINR of all users. Through extensive simulations, the proposed method has shown superior performance over the basic GA for various system parameters and excellent generalization abilities, and therefore it provides a new direction for designing effective heuristic algorithms to solve difficult combinatorial problems in wireless optimizations. Future work will extend this approach to enhance the real-time performance, robustness and security of FAS using LLM.

Appendix

A. Optimized Crossover Function

```
def crossover(parents, M):
   n_parents, n = parents.shape
   offspring = np.empty((M, n),
       dtype=parents.dtype)

   gene_freq = np.zeros((n, np.max(parents) +
       1), dtype=float)
   for i in range(n):
      unique, counts = np.unique(parents[:,
          i], return_counts=True)
      gene_freq[i, unique] = counts /
          n_parents

   for i in range(M):
      p1, p2 =
          parents[np.random.choice(n_parents,
          2, replace=False)]
      for j in range(n):
         if gene_freq[j, p1[j]] >
             gene_freq[j, p2[j]]:
            offspring[i, j] = p1[j]
         elif gene_freq[j, p1[j]] <
             gene_freq[j, p2[j]]:
            offspring[i, j] = p2[j]
         else:
            offspring[i, j] = p1[j] if
                np.random.rand() < 0.5 else
                p2[j]
   return offspring
```

Listing 4. Optimized crossover function.

B. Optimized Mutation Function

```
def mutation(population, N):
   mutated_population = population.copy()
   p, n = population.shape

   for i in range(p):
      if np.random.random() < 0.7 and
          selected_port > 1 and
          np.random.random() < 0.5:
         idx1, idx2 = np.random.choice(n,
             size=2, replace=False)
         mutated_population[i, idx1],
             mutated_population[i, idx2] =
             mutated_population[i, idx2],
             mutated_population[i, idx1]
      num_mutations = max(1,
          int(selected_port * 0.15))
      idxs = np.random.choice(n,
          size=num_mutations, replace=False)
      noise = np.random.uniform(-N/5, N/5,
          num_mutations)
      mutated_population[i, idxs] = np.clip(
         np.round(mutated_population[i, idxs]
             + noise),
         0, N - 1
      ).astype(int)
   return mutated_population
```

Listing 5. Optimized mutation function.

C. Optimized AutoPort Function

```
def select_ports(K, n, N, Pt, B, H, noise):
   port_sample = np.zeros((B, n), dtype=int)
   num_candidates = 5 # Number of candidate
       solutions to generate

   for i in range(B):
      best_rate = -np.inf
      best_ports = None
      port_norms = np.linalg.norm(H[i],
          axis=1)

      for _ in range(num_candidates):
         # Construct candidate solution with
             weighted random selection
         weights = port_norms /
             np.sum(port_norms)
         candidate = np.random.choice(N,
             size=n, replace=False, p=weights)

         # Local search improvement
         improved = True
         while improved:
            improved = False
            current_rate = sinr_balancing(n,
                K, H[i, candidate], Pt,
                noise_power)

            for j in range(n):
               for alt_port in range(N):
                  if alt_port not in
                      candidate:
                     temp_ports =
                         candidate.copy()
                     temp_ports[j] = alt_port
                     temp_rate =
                         sinr_balancing(n, K,
                         H[i, temp_ports],
                         Pt, noise_power)

                     if temp_rate >
                         current_rate:
                        candidate =
                            temp_ports.copy()
                        current_rate =
                            temp_rate
                        improved = True

         if current_rate > best_rate:
            best_rate = current_rate
            best_ports = candidate.copy()
```

```
        port_sample[i] = best_ports

    return port_sample
```

Listing 6. The optimized AutoPort function.

References

[1] Y. Shen et al., "Design and implementation of mmWave surface wave enabled fluid antennas and experimental results for fluid antenna multiple access," arXiv preprint, arXiv:2405.09663, May 2024.

[2] J. Zhang et al., "A novel pixel-based reconfigurable antenna applied in fluid antenna systems with high switching speed," *IEEE Open J. Antennas & Propag.*, vol. 6, no. 1, pp. 212-228, Feb. 2025.

[3] B. Liu, K.-F. Tong, K. K. Wong, C.-B. Chae, and H. Wong, "Programmable meta-fluid antenna for spatial multiplexing in fast fluctuating radio channels," *Optics Express*, vol. 33, no. 13, pp. 28898–28915, 2025.

[4] K.-K. Wong, A. Shojaeifard, K.-F. Tong, and Y. Zhang, "Fluid antenna systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1950–1962, Mar. 2021.

[5] K.-K. Wong, K.-F. Tong, Y. Shen, Y. Chen, and Y. Zhang, "Bruce Lee-inspired fluid antenna system: Six research topics and the potentials for 6G," *Frontiers in Communications and Networks*, vol. 3, pp. 1–31, Mar. 2022.

[6] W. K. New, K.-K. Wong, H. Xu, K.-F. Tong, and C.-B. Chae, "Fluid antenna system: New insights on outage probability and diversity gain," *IEEE Trans. Wireless Commun.*, vol. 23, no. 1, pp. 128–140, Jan. 2024.

[7] W. K. New, K.-K. Wong, H. Xu, K. F. Tong and C.-B. Chae, "An information-theoretic characterization of MIMO-FAS: Optimization, diversity-multiplexing tradeoff and q-outage capacity," *IEEE Trans. Wireless Commun.*, vol. 23, no. 6, pp. 5541–5556, Jun. 2024.

[8] Z. Chai, K.-K. Wong, K.-F. Tong, Y. Chen and Y. Zhang, "Port selection for fluid antenna systems," *IEEE Commun. Lett.*, vol. 26, no. 5, pp. 1180-1184, May 2022.

[9] J. Zou, S. Sun and C. Wang, "Online learning-induced port selection for fluid antenna in dynamic channel environment," *IEEE Wireless Commun. Lett.*, vol. 13, no. 2, pp. 313-317, Feb. 2024.

[10] I. Krikidis, A. K. Singh and K. Jamieson, "Optimizing reconfigurable antenna MIMO systems with coherent Ising machines," Proc. *IEEE Int. Conf. on Commun. (ICC)*, June 2024, pp. 1-5.

[11] C. Efrem and I. Krikidis, "Transmit and receive antenna port selection for channel capacity maximization in fluid-MIMO systems," *IEEE Wireless Commun. Lett.*, vol. 13, no. 11, pp. 3202-3206, Nov. 2024.

[12] L. Zhang, H. Yang, Y. Zhao, and J. Hu, "Joint port selection and beamforming design for fluid antenna assisted integrated data and energy transfer," *IEEE Wireless Commun. Lett.*, vol. 13, no. 7, pp. 1833–1837, Jul. 2024.

[13] C. Zhang, Y. Xu, S. Peng, X. Guo, X. Ou, H. Hong, D. He, and W. Zhang, "Fluid antenna-aided robust secure transmission for RSMA-ISAC systems," arXiv preprint, arXiv:2503.05515, Mar. 2025.

[14] C. Wang, G. Li, K.-K. Wong, Z. Li, D. W. Kwan, and C.-B. Chae, "Fluid antenna system liberating multiuser MIMO for ISAC via deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 23, no. 9, pp. 10879–10894, Sept. 2024.

[15] N. Waqar, K.-K. Wong, K.-F. Tong, A. Sharples, and Y. Zhang, "Deep learning enabled slow fluid antenna multiple access," *IEEE Commun. Lett.*, vol. 27, no. 3, pp. 861–865, Mar. 2023.

[16] C. He, Y. Lu, and W. Chen, B. Ai, K. -K. Wong and D. Niyato "Graph neural network enabled fluid antenna systems: A two-stage approach," *IEEE Trans. Veh. Technol.*, early access, doi: 10.1109/TVT.2025.3570319.

[17] Y. Zhang, H. Yin, W. Li, E. Bjornson, M. Debbah, "Port-LLM: A port prediction method for fluid antenna based on large language models," *arXiv preprint*, arXiv:2502.09857, 2025.

[18] C. Wang, K.-K. Wong, Z. Li, et al. "Large Language Model Empowered Design of Fluid Antenna Systems: Challenges, Frameworks, and Case Studies for 6G", arXiv preprint arXiv:2506.14288, 2025.

[19] F. Liu et al., "A systematic survey on large language models for algorithm design," arXiv preprint, arXiv:2410.14716, Oct. 2024.

[20] F. Liu, X. Tong, M. Yuan, and Q. Zhang, "Algorithm evolution using large language model," *arXiv preprint arXiv:2311.15249*, 2023.

[21] F. Liu et al., "Evolution of heuristics: Towards efficient automatic algorithm design using large language model," in *Proc. Int. Conf. Machine Learning*, 2024, pp. 32201–32223, Vienna, Austria, 21-27 July 2024.

[22] B. Romera-Paredes et al., "Mathematical discoveries from program search with large language models," *Nature*, vol. 625, no. 7995, pp. 468–475, Jan. 2024.

[23] R. Li, L. Wang, H. Sang, L. Yao, and L. Pan, "LLM-assisted automatic memetic algorithm for lot-streaming hybrid job shop scheduling with variable sublots," *IEEE Trans. Evol. Comput.*, early access, 2025.

[24] H. Ye, H. Xu, A. Yan, and Y. Cheng, "Large language model-driven large neighborhood search for large-scale MILP problems," in *Proc. Int. Conf. Machine Learning*, Vancouver, Canada, 13- 19 July 2025.

[25] Y. Yao, F. Liu, J. Cheng, and Q. Zhang, "Evolve cost-aware acquisition functions using large language models," in *Proc. Int. Conf. Parallel Problem Solving from Nature*, 2024, pp. 374–390, Hagenberg, Austria, 14-18 Sept. 2024.

[26] M. Schubert and H. Boche, "Solution of the multiuser downlink beamforming problem with individual SINR constraints," *IEEE Trans. Veh. Technol.*, vol. 53, no. 1, pp. 18–28, Jan. 2004.

[27] W. C. Jakes, *Microwave mobile communications*, Wiley, New York, 1974.

[28] "LLM4AD: Large Language Model for Algorithm Design," https://github.com/Optima-CityU/LLM4AD.

[29] S. Martello, and P. Toth, "Lower bounds and reduction procedures for the bin packing problem. Discrete applied mathematics,", vol. 28, no. 1, pp. 59–70, July 1990.

[30] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67–71, April 1989.

[31] A. Vaswani et al., "Attention is all you need," in *Proc. Conf. Neural Inf. Process. Syst.*, vol. 30, pp. 1–15, 2017.