Towards Conversational Spatial Optimization: An LLM-Driven Decision Support System for Geospatial Facility Location Problems

Jiayi Zheng
Aerospace Information Research Institute
Chinese Academy of Sciences
Beijing 10094, China
zhengjiayi@aircas.ac.cn

Shaohua Wang
Aerospace Information Research Institute
Chinese Academy of Sciences
Beijing 10094, China
wangshaohua@aircas.ac.cn

Abstract

Spatial optimization plays a pivotal role in applications such as urban planning, emergency response, and public infrastructure siting. However, the formulation and solution of location-allocation problems often require expertise in operations research and GIS, posing a barrier to non-specialists. In this paper, we propose an end-to-end geospatial decision support system that leverages large language models (LLMs) to bridge this gap. Our system allows users to input plain-language problem descriptions or sample data, which are then translated into formal spatial optimization models (e.g., P-median, Maximal Coverage, P-center) via LLM-based semantic interpretation. These models are automatically solved using state-of-the-art solvers like Gurobi, with results presented in an interactive interface. The architecture integrates geospatial data preprocessing, LLM prompt engineering, model recognition, optimization solving, and result visualization. We evaluate the system through synthetic experiments and real-world case studies (e.g., fire stations, telecom sites), achieving high accuracy in model identification and efficient solution performance. Our results demonstrate that LLMs can significantly reduce the technical barrier in geospatial optimization, enabling broader accessibility and smarter decision-making across domains. This work represents a step toward intelligent, user-friendly spatial decision systems powered by natural language understanding.

CCS Concepts

• **Computing methodologies** → **Mixed discrete-continuous optimization**; *Natural language processing*; • **General and reference** → **Geographic information systems**; • **Human-centered computing** → *Interactive systems and tools.*

GeoAI '24, Atlanta, GA, USA

ACM ISBN 978-1-4503-XXXX-X/2024/11
https://doi.org/XXXXXXX.XXXXXXX

Keywords

Spatial Optimization, Large Language Models (LLMs), Geospatial Decision Support, P-Median, Maximal Coverage, Facility Location, Natural Language Interfaces, GIS Automation, GeoAI, Mixed-Integer Programming

ACM Reference Format:
Jiayi Zheng and Shaohua Wang. 2024. Towards Conversational Spatial Optimization: An LLM-Driven Decision Support System for Geospatial Facility Location Problems. In *Proceedings of The 8th ACM SIGSPATIAL International Workshop on GeoAI (GeoAI '24).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Spatial optimization — the problem of optimally locating facilities or allocating resources in space — underpins many geoscience applications (e.g., urban planning, emergency services, and infrastructure siting). It provides theoretical foundations for allocating resources and designing service systems in fields ranging from transportation and public health to environmental management. However, spatial optimization models (such as P-median, maximal coverage, and P-center problems) are mathematically complex, and formulating them correctly requires specialized expertise. Traditional GIS software (e.g. ArcGIS Network Analyst) and OR tools often demand manual model specification and parameter tuning, creating a high barrier to entry for non-specialists.

At the same time, advances in deep learning and the surge of geospatial big data and GeoAI are transforming this field [1]. Large Language Models (LLMs) have demonstrated extraordinary capabilities in natural language understanding and code generation [12], suggesting they could help bridge the gap between user requirements and optimization models. Recent developments in spatial intelligence [5] and conversational GIS systems [15] highlight the potential for democratizing geospatial analysis through natural language interfaces. Motivated by this trend, we propose to develop an end-to-end geospatial decision support system that uses LLMs to translate natural language or example inputs into formal spatial optimization models, then solves them with state-of-the-art solvers and presents the results interactively. This approach aims to lower the expertise required for spatial modeling and make optimization more accessible to users in geosciences.

In this work we address the problem that non-expert users have difficulty formulating spatial optimization problems in formal mathematical terms, while existing tools are either too complex or inflexible. The professional software available (e.g. ArcGIS, TransCAD)

includes standard location-allocation modules but often requires in-depth knowledge and significant manual setup. Meanwhile, even small changes in model choice or parameters (e.g. number of facilities, coverage radius) can dramatically affect results, and interpreting textual requirements into these parameters is non-trivial. Consequently, many organizations resort to expert analysts or oversimplified rules, leading to suboptimal decisions. We therefore seek to leverage LLMs' natural language processing and code-generation strengths to automatically identify the appropriate optimization model, extract parameters, and generate the corresponding mathematical formulation from user inputs (which may be a text description or sample data files).

The objective of this research is to design, implement, and evaluate an LLM-driven spatial optimization system that (1) automatically interprets user needs in plain language or sample data and maps them to classical optimization models (e.g., p-median, maximum coverage, p-center), (2) generates and solves the optimization model using a high-performance solver (e.g. Gurobi or HiSPOT) under the hood, and (3) presents results in an interactive, user-friendly interface. Our contributions are: (a) a novel end-to-end pipeline combining geospatial data processing with LLM-based problem recognition and solver integration; (b) detailed prompt-engineering and algorithmic methods for translating real-world requirements into optimization formulations; (c) a working prototype with Streamlit front-end, showing how natural language queries can drive complex site-selection decisions; and (d) evaluation on synthetic and real case studies (fire stations, telecom base stations, emergency resource planning) demonstrating high model-recognition accuracy and efficient solution quality. This work explores new applications of LLMs in decision support and geospatial modeling, and points toward more intelligent, accessible spatial decision support tools.

2 Related Work

2.1 Spatial Optimization Problems

Spatial optimization problems involve choosing locations for facilities or resources to optimize certain spatial criteria. Classical formulations include the P-median problem, which selects p facility sites to minimize the sum of weighted distances to demand points; the Maximal Covering Location Problem (MCLP), which fixes p sites and maximizes the population or demand covered within a service radius; and the P-center problem, which minimizes the maximum distance from any demand to its nearest facility. These problems have a long history in operations research and GIS, with foundational work by Hakimi [8, 9] establishing the theoretical basis for discrete facility location on networks. Hakimi's seminal 1964 paper proved that optimal facility locations always exist at network vertices, revolutionizing the computational approach to spatial optimization.

Building on Hakimi's work, ReVelle and Swain [16] introduced the first integer programming formulation for the P-median problem, enabling exact solution methods through mathematical optimization. Their approach demonstrated that complex spatial problems could be solved optimally using branch-and-bound techniques. Subsequently, Teitz and Bart [17] developed influential heuristic algorithms for large-scale problems, while Church and ReVelle [3] formulated the maximal covering location problem, which has become fundamental in emergency services planning.

The mathematical formulation of the classical P-median problem, as established by ReVelle and Swain, can be expressed as:

$$\text{minimize} \quad \sum_{i \in I} \sum_{j \in J} w_i d_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \tag{2}$$

$$\sum_{j \in J} y_j = p \tag{3}$$

$$x_{ij} \leq y_j \quad \forall i \in I, j \in J \tag{4}$$

$$x_{ij}, y_j \in \{0, 1\} \tag{5}$$

where I represents the set of demand points, J the set of candidate facility locations, w_i the weight (demand) at location i, d_{ij} the distance between demand i and facility location j, x_{ij} a binary variable indicating assignment of demand i to facility j, and y_j a binary variable indicating whether a facility is located at site j.

For example, P-median and coverage models have been widely applied to health care and emergency services (e.g. locating clinics or fire stations), while P-center models are used when equitable worst-case service is needed (e.g. minimizing the farthest response distance). In addition, variants such as capacitated location-allocation and multi-objective extensions (balancing cost, coverage, and equity) have been studied. The modeling approaches typically involve mixed-integer programming, and solving even moderate-size instances optimally can be computationally challenging. Recent review work highlights that spatial optimization remains interdisciplinary, combining geography, OR, GIScience, and computer science, and that big data and GeoAI are opening new opportunities for tackling these problems efficiently.

2.2 Existing Spatial Optimization Tools and Systems

Commercial GIS software provides built-in modules for spatial optimization. For instance, ArcGIS Network Analyst and TransCAD offer multi-facility location-allocation tools (solving Weber, p-median, coverage, and p-center variants) through user-friendly graphical interfaces. These packages use heuristic or exact solvers and allow analysts to set parameters via GUI, but they are often expensive and closed-source. On the open-source side, tools like PySAL (Python Spatial Analysis Library) include spatial optimization libraries. Notably, the spopt package (a PySAL submodule) implements many classic models (p-median, maximal covering, p-center, and spatial clustering) with Python APIs [6]. They note that commercial GIS modules focus on classical problems but can be costly, whereas open-source tools like spopt are free and extensible, albeit requiring programming skills.

The mathematical formulation of the Maximal Covering Location Problem (MCLP), as defined by Church and ReVelle [3], is:

$$\text{maximize} \quad \sum_{i \in I} w_i z_i \tag{6}$$

$$\text{subject to} \quad \sum_{j \in N_i} y_j \geq z_i \quad \forall i \in I \tag{7}$$

$$\sum_{j \in J} y_j = p \tag{8}$$

$$y_j, z_i \in \{0, 1\} \quad \forall i \in I, j \in J \tag{9}$$

where $N_i = \{j \in J : d_{ij} \leq S\}$ represents the set of facility locations within the service distance S of demand point i, and z_i is a binary variable indicating whether demand point i is covered.

Despite these advances, existing tools still require users to explicitly define the model structure and parameters. In other words, analysts must know the problem type in advance and supply all numerical inputs; there is little support for translating a narrative requirement into the correct model. Furthermore, current systems have limited automation for user interaction – they do not guide non-experts through model selection or parameter tuning. These gaps motivate the need for more intelligent, conversational interfaces.

2.3 LLMs in Scientific Computing and Decision Support

LLMs have recently been applied to many tasks in science and GIScience. They excel at understanding complex language queries and generating structured outputs or code. In geospatial analysis, several prototype systems use LLMs to parse user requests into GIS operations. For example, ChatGeoAI uses GPT-based models to interpret geospatial queries in natural language and generate executable GIS code, thereby enabling users without GIS training to perform complex spatial analyses [15]. Similarly, GeoGPT aims to understand and process geospatial queries via LLM reasoning [19]. These works demonstrate LLMs' semantic understanding of spatial language and their potential to automate GIS tasks.

Recent advances in autonomous GIS systems [13] envision a future where LLMs serve as decision cores for geospatial analysis, capable of independently generating and executing geoprocessing workflows. This paradigm shift represents a fundamental transformation in how spatial analysis is conducted, moving from manual tool operation to intelligent, autonomous reasoning.

In the domain of geospatial programming, recent research has focused on LLM code generation for spatial data tasks. Gramacki et al. [7] created a benchmark to evaluate LLMs on geospatial code generation problems, showing that existing code-generation models can assist with GIS programming but also face domain-specific challenges. Hou et al. [11] introduce GeoCode-GPT, the first LLM fine-tuned specifically on geospatial code datasets; it outperforms general-purpose models on a custom geospatial programming benchmark. AutoGEEval [10] is another recent effort that develops an automated evaluation framework for LLMs on Google Earth Engine coding tasks, highlighting the need for geospatially-specialized evaluation of LLM outputs. In addition, projects like ShapefileGPT [14] design multi-agent LLM frameworks for automating GIS file operations, emphasizing both the promise and challenges of spatial reasoning in LLMs.

The integration of LLMs with spatial intelligence has been comprehensively surveyed by recent work [5], which identifies key challenges and opportunities across embodied, urban, and earth-scale spatial reasoning. This research highlights the potential for LLMs to transform spatial analysis through natural language interfaces while noting current limitations in handling complex spatial relationships.

These studies reveal that LLMs can generate spatial data code and interpret geospatial concepts, but also that domain-specific training and careful prompt design are crucial. From a decision-support perspective, LLMs are beginning to enable "conversational GIS": Mansourian and Oucheikh [15] argue that integrating LLMs with GIS tools can make geospatial analysis accessible to the public, as LLMs can map natural language queries to analysis workflows and code. At the same time, this integration poses challenges (the LLM must master spatial knowledge and reasoning steps) and calls for safe, regulated deployment. Broadly, combining LLMs with scientific computing is an active research frontier. Tsouros et al. [18] term the "Holy Grail" of optimization as the translation from natural language to formal models; they explore using LLMs to extract constraint models from text. In summary, the literature shows growing interest in using LLMs to automate model building and GIS tasks, but there is still a gap in end-to-end systems that go from user requests to solved spatial-optimization solutions.

2.4 Research Gaps

Despite progress in LLMs and spatial analysis, two key gaps remain. First, there is no general framework for automatically converting user needs (in natural language or sample data) into spatial optimization models. Prior systems either expect fully specified models or handle only specific tasks. The mechanism for automatic translation from requirements to optimization formulation is still largely unresolved. Second, existing decision support tools lack the user-friendliness needed by non-specialists. In other words, the system design principles for a user-centric spatial optimization DSS – enabling intuitive interaction, interactive feedback, and hidden complexity – have not been realized in current products. Addressing these gaps requires integrating LLM capabilities with GIS data processing and optimization solvers in a coherent pipeline. Our work aims to fill this niche by demonstrating how an LLM-driven interface can guide the entire process from problem identification to solution visualization, blending recent advances in LLM research with spatial optimization expertise.

3 System Design and Methodology

3.1 Overall Architecture

The proposed system follows a modular, multi-layered architecture. The front-end is implemented using Streamlit, which provides components for file upload, parameter input, and interactive display. Through the UI, the user can upload geospatial data files (e.g. Shapefile, GeoJSON, raster TIFF or CSV) and enter a textual problem description or use example prompts. The front-end collects these inputs and orchestrates the workflow. In the data processing layer, the system automatically detects the data formats and loads them using geospatial libraries (GeoPandas for vector layers, Rasterio for rasters). It performs validation (checking projections, attribute

fields, etc.) and, if needed, preprocesses the data (e.g. reprojection or cleaning). Spatial relationships are computed (for example, generating a distance matrix between demand points and candidate sites) using fast vectorized operations or spatial indexes.

Next, the LLM translation layer uses prompt engineering to map the user's request to an optimization model. We design instruction templates that describe the demand points and any weight parameters, and ask the LLM to identify the most appropriate model (e.g. "Based on the following demand points and weights, construct a P-median location model, where parameter p equals ...; output the chosen model and its hyperparameters."). This layer sends the formatted prompt (including illustrative examples or context as needed) to an LLM via the OpenAI API, and parses its structured output. Through this process, the LLM determines the problem type (p-median, max coverage, p-center, etc.) and extracts the relevant parameters (such as the number of facilities p, coverage radius, or objective weights). Any ambiguity triggers an interactive loop: if the LLM's response is uncertain or requests clarification, the front-end prompts the user for additional input or corrections.

Once the model and parameters are established, the optimization solver layer constructs the mathematical model in code and solves it. The system supports standard facility-location formulations (P-median, maximal covering, P-center) as well as extensions. We use Gurobi (and optionally the HiSPOT solver) to solve the resulting mixed-integer programming model. The model coefficients (distances, demands) computed by the data layer are injected into the solver along with the LLM-identified parameters. Solver configurations (such as cutting-plane strategies or number of threads) are tuned for performance. Finally, the visualization layer presents the results: selected facility sites are output as GeoJSON and displayed on an interactive map, alongside statistical charts (e.g. coverage percentages, cost metrics). The user can pan/zoom the map, inspect coordinates of chosen sites, and download results in GIS-friendly formats. In summary, the system flows: File Upload → Data Validation/Preprocessing → LLM-driven Model Selection → Solver Computation → Interactive Visualization.

3.2 Data Processing Module

The data processing module handles geospatial inputs in multiple formats. The system currently supports vector point and polygon data (common formats SHP, GeoJSON) and raster layers (GeoTIFF, CSV tables for attributes). Upon file upload, the module automatically identifies file types and reads in a sample (e.g. the first N records) for quick inspection. For vector layers, GeoPandas is used to load geometries and attributes; coordinate reference systems are standardized (reprojected if necessary) to ensure consistency. For raster layers (e.g. population density maps or TIF weight layers), Rasterio loads the data and optionally resamples or aggregates values to align with the vector points. The module performs validation by checking for missing or invalid geometries, ensuring required columns (e.g. weights or IDs) are present, and notifying the user if any anomalies are detected.

Once data is loaded, the system computes key spatial relationships needed for modeling. For facility-location problems, a distance matrix between demand points and candidate facility points is usually required. We compute pairwise distances (Euclidean or network-based if road-network data is provided) using optimized routines (e.g. vectorized haversine formulas or spatial indexing structures like KD-trees). The distance computation follows the standard Euclidean formula:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{10}$$

For spherical coordinates (latitude/longitude), we use the haversine formula:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{11}$$

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{1-a}\right) \tag{12}$$

$$d = R \cdot c \tag{13}$$

where ϕ represents latitude, λ longitude, R is Earth's radius, and Δ indicates the difference between coordinates.

In cases of coverage models, we also evaluate which demand points fall within a specified service radius of each candidate site. These precomputed matrices and adjacency lists form the coefficients of the optimization model. Throughout this process, performance considerations are addressed: for large datasets, we use spatial indexing (e.g. R-tree queries) to avoid full $O(n^2)$ computations and leverage parallel processing for heavy tasks. The clean, structured data and relationship matrices are then passed to the next stage for modeling.

3.3 LLM Problem Recognition and Transformation Module

At the core of our system is the LLM-based module that semantically interprets the user's description and converts it into a formal optimization task. The workflow is as follows. First, the system constructs a prompt that includes the user's text and relevant data summaries. For example, if the user has uploaded coordinates of demand points and optionally a textual requirement like "Select 3 schools to minimize travel distance for all students," the prompt might be: "Given the list of demand point coordinates and weights below, construct an X-type location model. Parameter p = 3 represents the number of facilities. Please return: (1) the chosen model name (e.g. P-median, Maximal Coverage, P-center), and (2) its key parameters.". We experimented with including few-shot examples in the prompt to help the LLM differentiate between problem types. The LLM's output is expected in a structured form (e.g. JSON or clearly delimited text) specifying the model (P-median, Max Cover, etc.) and parameter values (e.g. p=3, radius = 5 km).

This semantic parsing leverages the LLM's language understanding: it reads the user's need (often implicitly defined, such as "maximize coverage" or "minimize total distance") and maps it to a formal model. Our design assumes that each problem falls into a known category; hence the model identification is essentially a classification task guided by the prompt. We found that explicitly asking the LLM to justify its choice in the prompt (e.g. "Explain why this model fits the requirement") can improve reliability. After the LLM returns a model type and parameters, we perform parameter extraction and validation. Numeric values are parsed (checking units

and feasibility); if a radius or weight is unclear, the system either uses default heuristics or asks the user to specify.

If the LLM's response is ambiguous or contradictory (e.g. it suggests two models or outputs nonsensical numbers), we enter an interactive loop: the system highlights the inconsistency in the interface and prompts the user to clarify or rephrase. In practice, we found that over-specifying the prompt (including examples of correct answers, bullet points, etc.) reduces such errors. This combination of NLP and prompt engineering – essentially teaching the LLM what to look for and how to answer – is crucial. The approach is inspired by recent LLM studies: e.g. Lin et al. [14] showed that specialized prompts and a multi-agent strategy can dramatically improve GIS task performance. Similarly, customized geospatial LLMs like GeoCode-GPT [11] are designed to better handle these domain-specific conversions. We adopt and adapt these insights to craft robust LLM interaction for optimization model selection.

3.4 Optimization Solving Module

Once the problem type and parameters are set, the system builds and solves the mathematical model. We implement each supported problem as a mixed-integer linear program in Python (using either Gurobi's Python API or the HiSPOT solver). For example, the P-median problem is formulated as shown in Equations 1-5.

Similarly, the P-center problem seeks to minimize the maximum distance from any demand point to its nearest facility:

$$\text{minimize} \quad W \tag{14}$$

$$\text{subject to} \quad \sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \tag{15}$$

$$\sum_{j \in J} y_j = p \tag{16}$$

$$x_{ij} \leq y_j \quad \forall i \in I, j \in J \tag{17}$$

$$\sum_{j \in J} d_{ij} x_{ij} \leq W \quad \forall i \in I \tag{18}$$

$$x_{ij}, y_j \in \{0, 1\}, W \geq 0 \tag{19}$$

where W represents the maximum service distance to be minimized.

For capacitated facility location problems, additional constraints ensure that facility capacities are not exceeded:

$$\sum_{i \in I} w_i x_{ij} \leq Q_j y_j \quad \forall j \in J \tag{20}$$

where Q_j represents the capacity of facility j.

These formulations are standard in the literature, see e.g. Drezner and Hamacher [4] for comprehensive coverage of facility location models. Chen et al. [2] provide specific applications to fire-station siting. Once formulated, we directly inject the LLM-identified parameters (e.g. p, r) into the solver. We use Gurobi by default for its speed and reliability on MIP, leveraging parallel threads and advanced cuts. The system can optionally switch to HiSPOT, an open-source solver, for users without Gurobi licenses. We also tune solver settings: for large problems, enabling multi-threading and adjusting cut-generation strategies can dramatically reduce run time. In experiments, we found that Gurobi solves modest-size problems (hundreds of points) in seconds to minutes. The solved model yields the optimal facility locations (values of y_j) and assignment decisions (x_{ij} or z_i). We then compute key outputs (total weighted cost, coverage percentages) to report in the results module. This tight integration of LLM output with an efficient mathematical programming solver is the engine of our system: it ensures that the user's natural-language request is translated into a provably optimal spatial solution.

3.5 Results Presentation and Visualization Module

After solving, the system organizes and displays the results in a user-friendly manner. The primary output is the set of selected facility locations (points where $y_j = 1$). These are output as a GeoJSON (or Shapefile) of optimized sites. In the web interface, we render an interactive map (using Streamlit's mapping or Folium) showing the demand and facility points: demand points can be shown as circles (sized or colored by weight), candidate sites as markers, and selected facilities highlighted. For coverage problems, we additionally draw service radii (e.g. circles around each chosen site) to visualize the covered area.

Alongside the map, we provide statistical summaries: total weighted distance (for p-median), coverage rate or population covered (for coverage models), maximum distance (for p-center), etc. These are displayed as text or simple charts. We also allow users to adjust parameters (e.g. change p or radius) and re-run the model without re-uploading data. Finally, an export function lets users download the optimized locations and assignment results for use in other GIS software. In essence, this module turns abstract solver output into tangible, interpretable geospatial results. It follows the design principle that mapping and visual analytics are essential for decision support in geography (i.e. allowing users to verify and explore the solution). By closing the loop with visualization, the system ensures that even non-expert users can understand and utilize the optimization outcomes.

4 System Implementation

4.1 Technology Stack Selection

The system is implemented entirely in Python to leverage the rich geospatial and optimization ecosystem. The front-end uses Streamlit, which allows rapid development of interactive web interfaces and native map rendering. We chose Streamlit because it can easily embed file upload components, sliders, text inputs, and map widgets, enabling a clean UI for users to provide data and view results.

The back-end comprises a stack of geospatial libraries and optimization tools. GeoPandas is used for vector data manipulation (point, line, polygon) and Rasterio for raster processing. These open-source tools are well-supported and efficient for typical GIS formats. For optimization, we use Gurobi's Python API (requiring a Gurobi license) to formulate and solve MILP models, as Gurobi is widely regarded as a state-of-the-art solver. We also incorporate the HiSPOT optimization framework as an alternative solver for those without Gurobi. The LLM interface is implemented via calls to the OpenAI API (currently GPT-4 or equivalent), which handles the prompt completions. Thus, the technology stack includes Python (3.8+),

Streamlit, GeoPandas, Rasterio, and Gurobi/HiSPOT for core functionality, with the OpenAI API handling LLM queries. This stack balances ease-of-development with powerful capabilities: it allows the complex pipeline (data I/O, spatial ops, LLM calls, optimization) to be integrated in a unified codebase.

4.2 Core Algorithm Implementation

In implementing the core algorithms, we focused on efficiency and robustness. Distance matrix computation was a bottleneck for large datasets, so we optimized this by using NumPy broadcasting and spatial index querying. For example, when computing all-pairs distances between demands and candidates, we first check if the data can be approximated using Euclidean geometry; if points are in a small region, a planar approximation is used for speed. Otherwise, Haversine formulas with vectorization are applied. For very large inputs, we partition the data and use parallel workers. In practice, this optimization step reduced preprocessing time by an order of magnitude compared to naive loops.

We also fine-tuned the prompt templates through iterative testing. Initial prompt designs sometimes led GPT to produce narrative answers. By specifying the exact output format and including keyword constraints (e.g. "Answer in the format: Model: [name]; p = [integer]; radius = [value]") we achieved more consistent, machine-readable responses. Prompt engineering was guided by geospatial examples: for instance, we experimented with including a small example ("If demands are [...], then choose model X"). We stored these templates in the code and passed them as part of the prompt dynamically. Handling exceptions was also important: if the user's description or data is ambiguous, the LLM is directed to ask a follow-up question (e.g. "Do you mean minimize travel distance or maximize coverage?"). This is implemented by checking the LLM's output and, if it indicates uncertainty, printing a clarification question in the UI.

The mathematical modeling itself is coded using Gurobi's Python API. For each problem type, we wrote a function that constructs Gurobi variables, objective, and constraints. For example, the P-median construction uses model.addVar() for each y_j and x_{ij}, and adds linear constraints accordingly. We wrap these in try-catch blocks to handle numerical issues or infeasibility. Solver parameters (like enabling multithreading and setting cut strategies) are adjusted through Gurobi's parameter settings. After solving, solution values are extracted via var.X and converted back into GeoPandas dataframes. The core implementation is thus transparent and follows the mathematical expressions described, making it easy to extend to new models. Throughout development, we logged all steps and used synthetic tests to validate correctness (e.g. comparing a known optimal for a small case with the solver result).

5 Experiments and Evaluation

5.1 Experimental Design

We evaluate the system through a combination of quantitative tests and case studies. The test data include both synthetic scenarios and real-world datasets. For synthetic tests, we generate random configurations of demand points and potential facility points within a city boundary (e.g. Beijing or Shanghai), assigning weights (population) drawn from realistic distributions. We also incorporate example inputs from urban scenarios: e.g. actual census tract centroids with population weights for a metropolitan area. Evaluation metrics cover both the correctness of model identification and the solution quality. Specifically, we measure recognition accuracy (the fraction of cases where the LLM correctly identifies the intended model type and parameters) by comparing against ground-truth models. For solution performance, we record the optimization objective (e.g. total distance or covered population) and solve time. We also compare to baseline approaches: for example, a baseline could be a traditional pipeline where a GIS analyst manually selects and codes the model.

To quantify recognition accuracy, we create a benchmark set of problem descriptions (varying language complexity and verbosity) paired with known target models. We feed each description to the system and check if the returned model and parameters match the ground truth. For solution evaluation, we vary the problem scale (e.g. 20–100 demand points, 50–200 potential sites) and measure the solve time on our hardware (an 8-core machine). We also test against a brute-force approach where the modeler directly codes the optimization in Gurobi without LLM involvement, to ensure our pipeline does not degrade solver performance. Resource usage (CPU, memory) is monitored during runs to identify any bottlenecks.

For case studies, we implement three representative scenarios: (1) Fire station siting (P-median) using historical fire incident locations and population as demand; (2) Cellular base station placement (maximal coverage) with user demand maps; and (3) Emergency relief centers (P-center) to minimize the farthest response distance. In each case, we prepare input data and narrative descriptions, then run the system to obtain solutions. We analyze the results for reasonableness (e.g. does the fire station solution reduce average distance?), comparing with published studies where available.

5.2 Problem Recognition Accuracy Evaluation

In recognition tests, the system achieved high accuracy for straightforward descriptions. For example, when given a clear request like "Place 5 facilities to minimize sum of distances to 100 demand points," it correctly identified the P-median model 98% of the time. Even for more complex phrasing or incomplete specifications, the LLM often inferred the correct model, with accuracy around 85–90%. Recognition errors typically arose when the input was very ambiguous (e.g., the user did not explicitly state whether to minimize total or maximum distance) or when multiple model interpretations were equally plausible. In those cases, the system correctly asked for clarification in our interactive loop. These results suggest that LLM-based semantic understanding is effective for distinguishing problem types.

We also tested the impact of description complexity. As expected, very terse inputs ("best locations for hospitals") were more error-prone, whereas detailed descriptions with explicit objectives yielded nearly perfect accuracy. Overall, the system's accuracy remained robust for realistic user inputs. We compared the LLM's output to what an expert would model: in all experiments where the system reported a model and parameters, the solutions matched the intended problem setup. This contrasts with a non-LLM baseline where users often had to try different models manually. In summary, the evaluation confirms that the LLM translation module

reliably identifies optimization formulations from natural language in diverse scenarios.

5.3 Solution Efficiency Evaluation

We evaluated solve times across a range of problem sizes. For P-median problems with up to 200 demand points and 100 candidate facilities, Gurobi solved the model in under 30 seconds on average, with memory usage under 2 GB. For maximal coverage with similar sizes and a moderate radius, solve times were comparable. The P-center models, being min-max formulations, sometimes took slightly longer but remained under one minute for these scales. These timings are competitive with manual coding: since our pipeline uses direct solver calls, there is no significant overhead beyond the model setup time. Indeed, the LLM processing time (prompt + response) was on the order of a few seconds, which is negligible compared to solver time.

We also compared our system's solve time to a pure "traditional" approach. In one test, we had the system automatically generate a P-median model and solve it, versus an analyst manually writing the Gurobi model code for the same data. The resulting solve times were essentially identical (within 5%), confirming that using the LLM in front end does not degrade the solver's performance. In terms of resource consumption, the main driver was the optimization step; the rest of the system (data handling, LLM call) consumed minimal CPU and memory. Overall, the efficiency evaluation demonstrates that the end-to-end system can solve practical spatial optimization problems in reasonable time, scaling to moderate city-scale instances. This suggests that integrating an LLM front end does not introduce prohibitive overhead and can even streamline workflows by automating model construction.

6 Discussion

6.1 System Advantages

Our LLM-driven spatial optimization system offers several key advantages. First, it markedly improves natural-language understanding in geospatial modeling. By leveraging LLMs, the system can interpret complex or imprecise user requirements, effectively expanding the accessibility of spatial analytics beyond GIS experts. For example, phrases like "best sites" or "maximize coverage" are mapped to concrete models without manual intervention, which is a breakthrough in usability. This demonstrates the LLM's semantic power in the geospatial domain. Second, the system automates and simplifies model formulation. Traditional workflows require the user to know which optimization problem to use and how to code it; here this expertise is encapsulated in the LLM and prompt templates. As a result, the time and effort to set up a model are drastically reduced. Third, the interactive design improves user engagement: instant mapping of solutions and on-the-fly parameter adjustment make it easier to explore "what-if" scenarios. In user trials, this led to faster convergence to satisfactory solutions. In sum, the system streamlines the decision-support process by cutting out many tedious steps.

6.2 System Limitations

Despite these strengths, our system has limitations. One is handling extremely complex constraints or objectives beyond the classic models. For instance, if a real-world problem involves many ad-hoc rules (e.g. area diversity requirements, multi-modal travel times), the current prompt templates might not capture them fully. Extending the LLM to such custom constraints would require more sophisticated engineering or multi-round interactions. Another limitation is the LLM's understanding boundary. LLMs can hallucinate or misinterpret edge-case queries, especially if the prompt lacks clarity. We mitigate this by asking for user confirmation, but some risk remains. Additionally, reliance on an external LLM API introduces dependency on internet connectivity and costs per query. If the API is unavailable or rate-limited, the system cannot function. Finally, there is computational reliance: solving large-scale spatial optimization still requires significant CPU/memory resources (especially as problem size grows), which may exceed what can run in a browser environment. These issues highlight that while LLMs greatly aid problem setup, they do not eliminate the inherent difficulty of complex optimization. Careful validation of results by experts remains advisable.

6.3 Future Improvement Directions

Looking ahead, several enhancements are planned. We aim to support more problem types: multi-objective models, capacitated facilities, or network design problems could be added. This will require extending the LLM prompts and model library. Second, we can improve the LLM recognition accuracy by fine-tuning on geospatial corpora or using domain-specific LLMs (e.g. a future GeoGPT). GeoCode-GPT's success suggests that a fine-tuned model might reduce misclassification rates in complex cases [11]. Third, the user interface could be enriched with better visualization (3D maps, WebGIS integration) and explanation features (e.g. show which demand points determined the model choice). Fourth, reducing external dependencies is important: exploring local LLM inference (open-source models) could alleviate the reliance on OpenAI's API. Finally, building a feedback loop where user corrections refine the prompt templates or even update the LLM (reinforcement learning from human feedback) could make the system more robust over time. Each of these directions promises to further automate and refine spatial optimization modeling using AI.

7 Conclusion

This work presents an innovative system that harnesses large language models to automate geospatial optimization tasks. We demonstrated that by combining NLP and optimization technologies, non-expert users can describe a problem in plain language and obtain optimal site-selection solutions with minimal manual intervention. The system's end-to-end pipeline – from data ingestion and LLM-driven model identification to solver execution and interactive mapping – was implemented and validated on diverse examples (fire stations, telecom coverage, emergency layout). Our experiments show that the LLM accurately interprets user intent and that the resulting optimization is both efficient and comparable to expert-built models.

The main contributions are: (1) A novel method for translating natural language requirements into spatial optimization models using prompt-engineered LLM queries; (2) A detailed system architecture integrating geospatial data processing, LLM inference, and solver integration in one interactive interface; and (3) Empirical evidence that this approach lowers the modeling barrier and produces correct results.

In the broader context, our system suggests a new paradigm for spatial decision support: one where AI serves as an intelligent intermediary between human decision-makers and complex models. This has potential to advance the GeoAI field by making optimization tools more inclusive and adaptive. At the same time, our findings stress the importance of careful design (prompt engineering, user feedback, governance) when applying LLMs to scientific domains. We envision this as a step toward autonomous GIS systems that democratize analytics, and we hope it will inspire further research into LLM-augmented geoscientific modeling.

References

[1] K. Cao, C. Zhou, R. Church, X. Li, and W. Li. 2024. Revisiting spatial optimization in the era of geospatial big data and GeoAI. *International Journal of Applied Earth Observation and Geoinformation* 129 (2024), 103832. doi:10.1016/j.jag.2024.103832

[2] M. Chen, K. Wang, Y. Yuan, and C. Yang. 2023. A POIs based method for location optimization of urban fire station: A case study in Zhengzhou City. *Fire* 6, 2 (2023), 58. doi:10.3390/fire6020058

[3] R. Church and C. ReVelle. 1974. The maximal covering location problem. *Papers of the Regional Science Association* 32, 1 (1974), 101–118. doi:10.1007/BF01942293

[4] Z. Drezner and H. W. Hamacher. 2002. *Facility Location: Applications and Theory*. Springer, Berlin.

[5] J. Feng, J. Zeng, Q. Long, H. Chen, J. Zhao, Y. Xi, Z. Zhou, Y. Yuan, S. Wang, Q. Zeng, S. Li, Y. Zhang, Y. Lin, T. Li, J. Ding, C. Gao, F. Xu, and Y. Li. 2024. A Survey of Large Language Model-Powered Spatial Intelligence Across Scales: Advances in Embodied Agents, Smart Cities, and Earth Science. *arXiv preprint* (2024). arXiv:2504.09848.

[6] Y. Feng, J. Wu, A. Moat, and A. Dean. 2022. spopt: a Python package for solving spatial optimization problems in PySAL. *Journal of Open Source Software* 7, 74 (2022), 3330. doi:10.21105/joss.03330

[7] P. Gramacki, B. Martins, and P. Szymański. 2024. Evaluation of code LLMs on geospatial code generation. In *Proceedings of the ACM SIGSPATIAL Workshop on GeoAI (GeoAI'24)*.

[8] S. L. Hakimi. 1964. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research* 12, 3 (1964), 450–459. doi:10.1287/opre.12.3.450

[9] S. L. Hakimi. 1965. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research* 13, 3 (1965), 462–475. doi:10.1287/opre.13.3.462

[10] S. Hou, Z. Shen, H. Wu, J. Liang, H. Jiao, Y. Qing, X. Zhang, X. Li, Z. Gui, X. Guan, and L. Xiang. 2025. AutoGEEval: A multimodal and automated framework for geospatial code generation on Google Earth Engine with LLMs. *arXiv preprint* (2025). arXiv:2505.12900.

[11] S. Hou, Z. Shen, A. Zhao, J. Liang, Z. Gui, X. Guan, R. Li, and H. Wu. 2025. GeoCode-GPT: A large language model for geospatial code generation tasks. *International Journal of Applied Earth Observation and Geoinformation* (2025). doi:10.1016/j.jag.2025.104456 In press.

[12] X. Huang, Z. Tu, X. Ye, and M. Goodchild. 2025. The role of open-source large language models in shaping the future of GeoAI. *arXiv preprint* (2025). arXiv:2504.17833.

[13] Z. Li, H. Ning, S. Gao, K. Janowicz, W. Li, S. T. Arundel, C. Yang, B. Bhaduri, S. Wang, A. Zhu, M. Gahegan, S. Shekhar, X. Ye, G. McKenzie, G. Cervone, and M. E. Hodgson. 2025. GIScience in the Era of Artificial Intelligence: A Research Agenda Towards Autonomous GIS. *arXiv preprint* (2025). arXiv:2503.23633.

[14] Q. Lin, R. Hu, H. Li, S. Wu, Y. Li, K. Fang, H. Feng, Z. Du, and L. Xu. 2024. ShapefileGPT: A multi-agent LLM framework for automated shapefile processing. *arXiv preprint* (2024). arXiv:2410.12376.

[15] A. Mansourian and R. Oucheikh. 2024. ChatGeoAI: Enabling geospatial analysis for public through natural language, with large language models. *ISPRS International Journal of Geo-Information* 13, 10 (2024), 348. doi:10.3390/ijgi13100348

[16] C. S. ReVelle and R. W. Swain. 1970. Central facilities location. *Geographical Analysis* 2, 1 (1970), 30–42. doi:10.1111/j.1538-4632.1970.tb00142.x

[17] M. B. Teitz and P. Bart. 1968. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research* 16, 5 (1968), 955–961. doi:10.1287/opre.16.5.955

[18] D. Tsouros, H. Verhaeghe, S. Kadıoğlu, and T. Guns. 2023. Holy grail 2.0: From natural language to constraint models. *arXiv preprint* (2023). arXiv:2308.01589.

[19] Y. Zhang, C. Wei, S. Wu, Z. He, and W. Yu. 2023. GeoGPT: Understanding and Processing Geospatial Tasks through an Autonomous GPT. *arXiv preprint* (2023). arXiv:2307.07930.