Incorporating domain knowledge into Memetic Algorithms for solving Spatial Optimization problems

Subhodip Biswas
Fanglan Chen
Department of Computer Science
Virginia Tech
{subhodip,fanglanc}@vt.edu

Zhiqian Chen
Department of Computer Science & Engineering
Mississippi State University
zchen@cse.msstate.edu

Chang-Tien Lu
Naren Ramakrishnan
Department of Computer Science
Virginia Tech
{ctlu,naren}@cs.vt.edu

ABSTRACT

Spatial optimization problems (SOPs) are characterized by spatial relationships governing the decision variables, objectives and/or constraint functions. These are mostly combinatorial problems (NP-hard) due to the presence of discrete spatial units. Hence, exact optimization methods cannot solve them optimally under practical time constraints, especially for large-sized instances. Motivated by this challenge, we explore the use of population-based metaheuristics for solving SOPs. To this end, we observe that the search moves employed by these methods are suited to real-parameter continuous search space rather. To adapt them to the SOPs, we explore the role of domain knowledge in designing spatially-aware search operators that can efficiently search for an optimal solution in discrete search space while respecting the spatial constraints. These modifications result in a simple yet highly effective spatial hybrid metaheuristic called SPATIAL, which is applied to the problem of school boundary formation (also called school redistricting). Experimental findings on real-world datasets reveal the efficacy of our algorithm in obtaining superior quality solutions in comparison to traditional baseline methods. Additionally, we perform an in-depth study of the individual components of our framework and highlight the flexibility of our method in assimilating other search operators as well as in adapting it to related SOPs.

CCS CONCEPTS

• **Theory of computation → Optimization with randomized search heuristics**.

KEYWORDS

Metaheuristic, domain knowledge, spatial optimization

ACM Reference Format:
Subhodip Biswas, Fanglan Chen, Zhiqian Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2020. Incorporating domain knowledge into Memetic Algorithms for solving Spatial Optimization problems. In *28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20), November 3–6, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3397536.3422265

SIGSPATIAL '20, November 3–6, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-8019-5/20/11…$15.00
https://doi.org/10.1145/3397536.3422265

1 INTRODUCTION

Solving a SOP involves search for an optimal configuration of a set of discrete spatial units such that some predetermined planning criteria and/or objectives are satisfied. It can also be defined as the use of mathematical/computational techniques for finding solutions to geographic decision problems under design constraints [24]. The optimization variables relate to the decision being made (as quantified by the objective function) while the constraints impose a set of necessary design considerations (specific to the studied problem) to be satisfied. The functions and constraints usually have spatial properties/topological relationships, including adjacency, contiguity (connectivity), intersection, similarity (distance), shape (compactness), containment, and so on, encapsulated in them [39]. Due to the interdisciplinary nature of SOPs, they have been studied in different contexts - location sciences [15, 23, 41], regionalization [6, 22], spatial data mining [16, 26], territory design [19, 36, 42], etc. A key feature of these SOPs is the presence of an underlying graph representation for encoding the spatial relationships as shown in Figure 1. Accordingly, a SOP is classified as either an assignment or a partitioning problem, with or without spatial constraints [40].

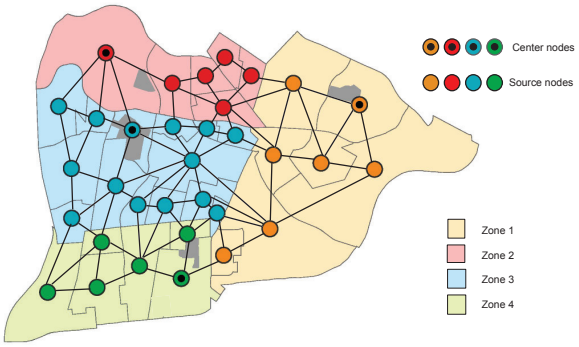


Figure 1: Graph-based representation of a geographical region formed by spatial units. The nodes encode the spatial units with edges linking the adjacent spatial units.

The discrete nature of a SOP makes them computationally challenging, i.e., an increase in the problem size leads to a combinatorial explosion of search space - the phenomenon where the computing time required to find the optimal solution of a NP-hard problem increases as an exponential function of the problem size [8, 25]. As a result, traditional optimization techniques are unable to optimally solve the problem under practical time constraints [14]. This has led to an increased adoption of metaheuristics for solving mid- to

large-sized combinatorial optimization problems [2]. In this work, we devise a population-based metaheuristic for solving SOPs inspired by the emerging field of Swarm Intelligence[1]. These methods instantiate search moves the closely mimic the complex social behavior of animals such as ant colonies, beehives, bird flocks, and so on, which may lead to an improved exploration of the decision or search space [5].

However, adapting methods based on Swarm Intelligence for solving SOPs has its own set of challenges: **(1) Real-parameter search move:** These algorithms employ linear search moves for perturbing existing solutions in order to explore the decision space. While these moves are ideal for solving real-parameter continuous optimization problems, they are hardly suited to the discrete search space present in SOPs. **(2) Spatial constraints:** SOPs are highly constrained in nature due to the presence of spatial relationships. Hence, the number of infeasible solutions surpass the number of feasible solutions significantly with an increase in problem size. This often results in expending tremendous computational effort in finding a feasible solution, especially when the search operators are not spatially cognizant. **(3) Exploration-exploitation trade-off:** Population-based models have a tendency for strong exploration during the initial stages of a trial run. This is detrimental to preserving solution structure and spatial constraints present in SOPs. **(4) Random (re)initialization:** Solutions are reinitialized at the start of a run or when they become infeasible. This practice has little value in terms of search efficiency, especially when the decision space is highly constrained locally. The newly generated random solution might be worse of than its predecessors and incur significant time in finding promising region of the earch space.

In view of these challenges, we devise a simple, easy-to-use search framework for solving SOPs inspired by the Artificial Bee Colony (ABC) algorithm [20]. The linear search moves of the original ABC algorithm are modified to perform spatially-aware search with provision for solution repair. The resulting hybrid metaheuristics, commonly known as memetic algorithm [29], is called **S**warm-based s**PA**atial meme**TI**c **AL**gorithm (**SPATIAL**) and is tested on the problem of school boundary formation [1, 35], a well-known SOP encountered in zone design. It involves the delineation of the public school attendance zones (boundaries) in a school district based on a series of factors, including operational efficiency, stability, geographic proximity, accessibility, contiguity, transportation cost, and so on. The main contributions of our work are as follows:

- We highlight the role of domain knowledge in designing spatially-aware search moves in order to make population-based metaheuristics work on SOPs. This is a fundamental contribution to the field of spatial optimization. Additionally, we rigorously test the new search moves on the problem of school boundary formation by extensive experimentation on two real-world geospatial datasets.
- We depict how SOPs can be encoded in the form of graphs and derive a mathematical formulation for the problem of school boundary formation. The formulation is general and can easily be extended to other similar spatial problems such as territory design, service redistricting, regionalization, etc.
- We demonstrate how spatial search operators take advantage of searching for solutions at the boundary between feasible and infeasible decision space, especially in exploring the previously unseen solutions. This is an important finding in context of combinatorial optimization probelms.
- We empirically verify how population-based metaheuristics can benefit from parallel architecture of the computing platform in carrying out the search without noticeable increase in computational time.

[1]Swarm Intelligence is defined as the study and design of computational optimization techniques based on the collective intelligence emerging from a large population of search agents with simple behavioral patterns for communication and interaction [3].

2 BACKGROUND

2.1 Metaheuristics

The term *metaheuristic* was coined by Glover [9] and refers to a set of high-level problem-independent instructions or strategies for developing heuristic optimization algorithms [38]. Metaheuristics fall under the sub-field of stochastic optimization problems. As the name implies, these methods employ randomness or stochasticity for finding solutions to computationally hard problems in limited time. Exact (traditional) methods of mixed-integer optimization, like dynamic programming and branch-and-bound, are prone to combinatorial explosion. This is why metaheuristics are often adopted as a viable alternative to exact methods for solving computationally difficult problems, especially when the size of the problem instance is large and/or there is a practical time constraint.

Metaheuristics are classified into three broad classes depending on how the solutions (or their representations) are manipulated. **1) Local search metaheuristics** rely on iterative improvement by making small perturbations to a single solution. **2) Constructive metaheuristics** build solutions from constituent elements by adding them one at a time. **3) Population-based metaheuristics** start with an initial set of solutions and improve them in an iterative manner by selecting and combining existing solutions. Henceforth, we shall focus on population-based models and up next discuss how they can be adapted to solve SOPs.

2.2 Population-based models

These methods can be further subdivided into two groups: *a) Evolutionary algorithms* (EAs) are global optimization methods that mimic the process of natural evolution, by maintaining a population of solutions and improving them via the process of recombination and selection, till a locally optimal solution. EA methods are studied under two broad umbrella - Evolutionary Computation (EC) and Swarm Intelligence (SI) - based on the inspiration behind the modeling [5]. *b) Deterministic methods* include methods like scatter search and path relinking that follow a set of deterministic rules in selecting, generating and updating new solutions [12].

Recently, a line of work has focused on assimilating ideas from different classes into a "hybrid" framework. One such framework, which combines recombination operator (from EAs) with local search, is called *memetic algorithm* [28]. Memetic algorithms benefit from the synergy between iterative improvement of the local search and the recombination operation of the population-based methods. We take a step in this direction by proposing a memetic algorithmic framework for solving SOPs inspired by the ABC algorithm, a

continuous-valued real-parameter global optimization algorithm based on the foraging behavior of honeybees [20]. For more details about the ABC algorithm kindly refer to Part A of the Appendix. Our approach differs from the ABC algorithm in the terms of the spatially aware search moves based on domain knowledge.

2.3 Domain knowledge for spatial awareness

Domain knowledge is the set of auxiliary information that a metaheuristic needs to be able to efficiently search for a feasible solution. It includes both *model-specific* and *problem-specific* instructions.

In population-based models like ABC, given an initial population of solutions (parents), a linear search move is employed to produce a new set of solutions (offsprings). These linear moves are not suited to discrete search space. To add to the difficulty, the presence of spatial constraints (topological properties), like contiguity, in SOPs makes it harder to find feasible solutions. Even the traditional constraint handling techniques used in conjunction with EAs are of little help [27]. It becomes increasingly impracticable for EAs with linear search operators to solve SOPs [17]. Hence, domain knowledge is found to be helpful in integrating spatially-aware local search techniques within the EA framework.

We start with a set of initial feasible solution and then improve the solutions in an iterative manner by moving spatial units between the neighboring zones [30, 31]. Two types of moves are possible: a) *move one unit* at a time; b) *swap units* between two neighboring zones. The swaps are permitted only if it leads to an improvement in the objective function. Even then, such local moves may result in breaking spatial contiguity of the involved zones, thereby leading to an infeasible solution. Additionally, the greedy nature of the moves may prevent exploring the decision space beyond the immediate neighborhood of the incumbent solution. In such scenarios, path linking can help in repairing the solutions if they enter the infeasible search space [13]. However, we must first allow the solutions to reach the infeasible region before repairing them back to feasibility. This causes the solutions to undergo strategic oscillations between the feasible and infeasible search space, and often results in better solutions [10]. The role of problem-specific domain knowledge will be discussed in the subsequent sections.

3 SPATIAL OPTIMIZATION PROBLEM

Graphs provide a flexible way of encoding the spatial relationship between spatial entities and can easily be used in an optimization setting [7]. In this section, we outline general graph-based representations underlying in most of the SOPs and follow up with a real-life instance of the problem – school boundary formation. The proposed formulation is flexible enough to be extended (with some modifications) to similar spatial problems.

3.1 Graph-based representation

Let us assume a geographical region is composed of smaller-sized spatial (areal) units, which can be represented as nodes in a graph. An edge connects two nodes if their respective spatial units share a common boundary (commonly called rook contiguity in spatial sciences). Such a graph-based representation of a geographic region was shown earlier in Figure 1. Next, we describe the graph notations and use it to define an SOP.

Let a geographical region be composed of N spatial units and is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of nodes and $\mathcal{E}$ is the set of edges connecting adjacent nodes. Here, $\mathcal{G}$ is commonly called *contiguity graph* and is a simple planar graph with the nodes encoding the spatial entities and the edges capturing the spatial adjacency relationship between the entities. Alternatively, a node v can be represented by its index, i.e., $v \in \{1, 2, \ldots, N\}$. These nodes may have their attributes, i.e., $\mathbf{\Pi}(\mathcal{G}) = \{\pi_1, \pi_2, \ldots, \pi_N\}$, where π_v indicates the node attribute set corresponding to node v. Let π_v is represented by a tuple

$$(L_v, P_v),$$

where $L_v = [(x_1, y_2), (x_2, y_2), \ldots, (x_t, y_t), (x_1, y_1)]$ is the list of geographic coordinates (latitude-longitude) that define the boundary polygon of the v^{th} spatial unit, and P_v is the vector of feature values corresponding to it. Besides, there exists a similarity matrix $\mathbf{W}(\mathcal{G}) = (W_{uv})_{u=1,\ldots,N\ v=1,\ldots,N}$ that captures the relationship between any pair of nodes. Popular choices for the similarity metric include the distance function or the binary adjacency relationship. Similarity can also be considered as an edge attribute since it is generally defined for edges connecting adjacent nodes.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ corresponding to a geographical region, a SOP involves search for a feasible (often locally optimal) solution $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, such that the spatial configuration of $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$ satisfies pre-defined problem criteria/constraints while minimizing certain objective(s). We are concerned with the zoning problem, where $\mathcal{E}' \subset \mathcal{E}$ and $\mathcal{V}' = \mathcal{V}$ such that the nodes in $\mathcal{V}'$ are partitioned into subgraphs called *zones*.

3.2 School boundary formation (SBF) problem

The public school system, in countries like the US, operates through the functioning of a *school district*, which represents a large geographical region coterminous with the boundary of a county, subdivision, or a city. These districts are administrative units for overseeing the local jurisdiction of public schools. A school district is composed of smaller spatial units called planning zones or student planning areas (SPAs). These SPAs are grouped to form a geographically contiguous area, called the *school attendance zone* (SAZ), which defines the boundary of a school. The schools at different levels (elementary, middle, and high) have a well-defined boundary often arranged in a hierarchical manner. In a school district, the rule of thumb is that students attend the school assigned to their residing SPA. In Figure 2, we illustrate a school district with its constituent - school locations, SPAs, and SAZs. We see that the school district data has an underlying graphical structure. From henceforth, we shall use the following groups of terms interchangeably: nodes/spatial units/SPAs and subgraphs/zones/SAZs.

These school boundaries (SAZs) are often redrawn due to constantly shifting demographics within the school district which in turn leads to closing, opening, or building modification of schools. During a school boundary process, the planning department of the school district may consider many factors such as program capacity, accessibility, proximity, presence of man-made/geographical barriers, transportation costs, etc. Though there is no specified order in which these factors will be considered, the school boundaries are mostly modified to balance the present student enrollment with the schools' program capacity.

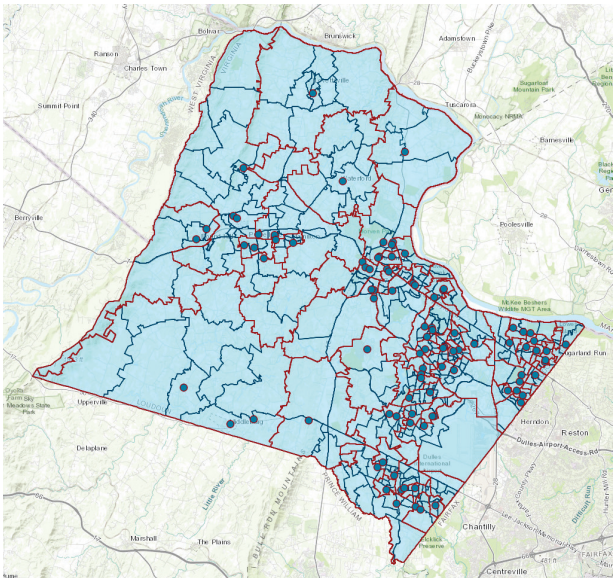


Figure 2: A GIS visualization showing the school district of Loudoun county, VA. The smaller polygons (with blue-colored border) are the SPAs and larger polygons (with brown-colored border) are the SAZs of elementary schools. The blue dots indicate the locations of all the public schools in the county.

We consider the naturally quantifiable factors – student enrollment to program capacity (capacity utilization) and school proximity (compactness) – as soft constraints, and spatial factors – geographic contiguity[2] and hard partition – as hard constraints. A feasible solution should satisfy all the hard constraints. On the other hand, the soft constraints can be violated with problem objective encoding the degree of violation. The optimization problem involves determining a feasible solution with the minimum possible (soft) constraint violation.

In instantiating the problem, let the SPAs be represented as

$$P_v = \left({}_{\text{ES}}n_v, {}_{\text{MS}}n_v, {}_{\text{HS}}n_v, {}_{\text{ES}}c_v, {}_{\text{MS}}c_v, {}_{\text{HS}}c_v\right), \quad v \in \{1, 2, \ldots, N\}, \quad (1)$$

where ${}_{\text{L}}n_v$ is the student population residing in SPA i corresponding to the school level L (ES: elementary school, MS: middle school, HS: high school), and ${}_{\text{L}}c_v$ is the program capacity of the schools contained in the same SPA. For majority of the SPAs that don't enclose any school inside them, we have ${}_{\text{ES}}c_v = 0$, ${}_{\text{MS}}c_v = 0$, ${}_{\text{HS}}c_v = 0$. We assume that all the schools in a school district follow a consistent grade structure with respect to the school levels.

Problem definition: If a school district is composed of N SPAs and K schools[3], and can be represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that $|\mathcal{V}| = N$ and $|\mathcal{E}| \leq 3N - 6$ (when $N \geq 3$ as per Euler's polyhedral formula), the solution to the SBF problem is a spatial configuration of K connected subgraphs (contiguous SAZs) such that it minimizes a weighted objective of capacity utilization and compactness (proximity). It can be mathematically written as

$$\underset{\mathbf{S}}{\text{minimize}} \quad \mathcal{F}(\mathbf{S}) \quad (2a)$$

s. t.

$$\sum_{z=1}^{K} S_{zv} = 1, \qquad \forall v = 1, 2, \ldots, N, \quad (2b)$$

$$\sum_{v=1}^{N} T_{zv} = 1, \qquad \forall z = 1, 2, \ldots, K, \quad (2c)$$

$$T_{zv} \leq S_{zv}, \qquad \forall v = 1, 2, \ldots, N, \ \forall z = 1, 2, \ldots, K, \quad (2d)$$

$$\sum_{v \in \mathcal{N}(A) \setminus A} S_{zv} - \sum_{v \in A \cup \{l\}} S_{zv} \geq 1 - (|A| + 1),$$
$$\forall z = 1, 2, \ldots, K, \ \forall l = 1, 2, \ldots, N,$$
$$\forall A \subset \{1, 2, \ldots, N\} \setminus \mathcal{N}(\{l\}) \neq \emptyset, \quad (2e)$$

$$T_{zv}, S_{zv} \in \{0, 1\}, \qquad \forall v = 1, 2, \ldots, N, \ \forall z = 1, 2, \ldots, K. \quad (2f)$$

where,
$v = \{1, 2, \ldots, N\}$: is the index corresponding to the nodes;
$z = \{1, 2, \ldots, K\}$: is the index corresponding to the subgraphs;
$\mathbf{S} = (S_{zv})_{z=1,2,..,K\, v=1,2,..,N}$: is a binary matrix $\{0, 1\}^{K \times N}$, where S_{zv} is 1 if node v is assigned to subgraph z, otherwise 0;
$\mathbf{T} = (T_{zv})_{z=1,2,..,K\, v=1,2,..,N}$: is another binary matrix $\{0, 1\}^{K \times N}$, where T_{zv} is 1 if node v is the center of subgraph z, otherwise 0;
$\mathcal{I}_z = \{v | S_{zv} = 1\}$: is the indexset containing the indices of nodes present in subgraph z.

Constraints (2b), (2c), and (2d), ensure that each node can be assigned to one subgraph only and each subgraph can have at most one center node, respectively. The connectivity of each subgraph is ensured by constraint (2e), where $\mathcal{N}(V)$ refers to all the nodes either represented by V or adjacent to some node represented by V. This constraint checks whether each subgraph (zone) is connected (contiguous) with respect to its constituent nodes (spatial units) and performs an exponential number of comparisons of order $O(KN2^N)$. Due to this combinatorial explosion of the constraints, it becomes impracticable for exact optimization methods to solve mid- to large-sized instances of this problem.

The problem defined in Equation (2) is a constrained optimization problem with binary decision variables, i.e., matrices $\mathbf{S}$ and $\mathbf{T}$. In these problems, domain knowledge is often used in prefixing a subset of nodes as centers of their respective zone. Here, we can consider the SPAs with schools inside them as center nodes, denoted by $\overline{\mathcal{V}}$, and accordingly fix the variable $\mathbf{T}$ resulting in a reduced number of decision variables and constraints in the optimization problem. Now, we solve a constrained assignment problem by minimizing a non-linear objective function $\mathcal{F}(\mathbf{S})$ computed as the total violation of soft constraints (aspatial and spatial):

$$\mathcal{F}(\mathbf{S}) = \lambda \underbrace{\sum_{z=1}^{K} \left| 1 - \frac{\sum_{v=1}^{N} S_{zv} \cdot {}_{\text{L}}n_v}{\sum_{v=1}^{N} S_{zv} \cdot {}_{\text{L}}c_v} \right|}_{\text{capacity utilization (aspatial)}} + (1 - \lambda) \underbrace{\sum_{z=1}^{K} \left| 1 - \frac{4\pi \cdot \text{Area}(\mathcal{I}_z)}{\text{Perimeter}^2(\mathcal{I}_z)} \right|}_{\text{boundary compactness (spatial)}}, \quad (3)$$

where, both the components asymptotically approaches 0 and are linearly weighted. $\lambda \in (0, 1)$ is a weighing factor that judges the relative importance of these objectives. We prefer to use λ such that

[2] One can travel from one part of the SAZ to another part without leaving the SAZ.
[3] The number K depends on the school level L := ES, MS or HS, since we solve the problem at each level independently.

$\lambda/(1-\lambda) \geq 2$ since capacity utilization is the predominant reason behind changing school boundaries.

The objective function in (3) generalizes to any SOP seeking a $K-$partition of a geographical region by minimizing a similarity-based dispersion metric while being subjected to balance and contiguity constraints. Here, we preferred shape compactness (non-linear) over distance-based dispersion (linear) due to the arbitrary locations of schools and lack of clustering structure in the dataset. Similar problems with slightly differing constraints and objectives are often encountered in designing commercial territories [34].

4 OUR APPROACH

The SPATIAL technique solves a SOP by maintaining a population of trial solutions. It starts by initializing the trial solutions. Then, the solutions are improved in an iterative manner (till a termination criterion is met) by executing two types of search moves: *local improvement* and *spatially-aware recombination.*

4.1 Initialization

Let $\mathcal{X} = \{\mathbf{S}^{(1)}, \mathbf{S}^{(2)}, \ldots, \mathbf{S}^{(SN)}\}$ represent a population of trial solutions of size SN. Each solution represents a different assignment of spatial units based on the binary matrix $\mathbf{S}^{(i)}$. We do not use the matrix $\mathbf{T}^{(i)}$ in solution representation since it has already been prefixed using domain knowledge. To ensure each solution is initially feasible, we adopt a two-step approach as elucidated below.

Seeding. It identifies a set of seed nodes, each of which is assigned to an unique subgraph. The problem-specific domain knowledge helps to select the seed nodes and is set equal to the number of zones we wish to find. Alternatively, we may apply the $K-$mediods algorithm to identify centrally located nodes and designate them as seed nodes. In our problem setting, the seed nodes correspond to the center nodes, i.e., the SPAs containing schools inside them. This results in forming K partial zones with one spatial unit in each zone. This ensures satisfaction of the constraints (2c) and (2d).

Contiguity-based growth. Following the seeding process, we apply the growth phase to grow the zones in a guided manner. A zone is randomly selected and grown by adding an adjacent unassigned spatial unit to it. The process is repeated till all the spatial units have been assigned. This results in a partition of K contiguous zones (connected subgraphs) satisfying constraints (2b) and (2e).

The pseudocode of initialization steps is given in Algorithm 1.

Algorithm 1: Initialization

```
Input  : G : Contiguity graph, SN : Population size, L: School level.
Output : X : Population of trial solutions
begin
    Determine the center nodes V̄ for school level L and set K ← |V̄|
    X ← {}                                              ▷ Empty population
    for i = {1, 2, ..., SN} do
        V: Get the set of nodes in G
        Seeding phase ▷
        Set z ← 0 and S^(i) as a zero matrix O_{K×N}
        for v ∈ V̄ do
            z ← z + 1
            S^(i)_zv ← 1                                    ▷ Assignment
            V ← V \ {v}
        Guided growth phase ▷
        do
            z: Randomly pick a zone from {1, 2, ..., K}
            N(z) : Find unassigned nodes adjacent to zone z
            while |N(z)| > 0 do
                v: Randomly select a node from N(z)
                S^(i)_zv ← 1                                ▷ Assignment
                N(z) ← N(z) \ {v} , V ← V \ {v}
        while |V| > 0
        X ← X ∪ {S^(i)}
return X
```

4.2 Local improvement

To explore the neighborhood of each candidate solution, we apply the local search move. In optimization literature, this is similar to the notion of search space exploitation. We sequentially scan each solution $\mathbf{S}^{(i)}$, $i = \{1, 2, \ldots, SN\}$, and select any random zone z, $z \in \{1, 2, \ldots, K\}$ within it for local improvement by moving any adjacent spatial unit (sharing boundary with zone z but belonging to a different zone w, $w \in \{1, 2, \ldots, K\} \setminus \{z\}$) into zone z. If the move results in violation of spatial constraints either in zone z or in zone w, the solution becomes infeasible. We deal with the spatial constraint of contiguity in a zoning problem. Thus, a solution with disconnected zones in it is infeasible and needs to be brought back into the feasible search space by applying a repair mechanism.

To repair a solution, we start by enumerating the connected components in the disconnected zone, say z, using breadth-first search (BFS) traversal. Then, each connected component is analyzed; if it doesn't contain a seed node, we reassign it to the neighboring zones randomly. In case there is no prior information about the center nodes, we can retain merely the largest-sized connected component of zone z while reassigning the remaining ones. The repaired solution $\tilde{\mathbf{S}}^{(i)}$ might be few steps away from the incumbent solution $\mathbf{S}^{(i)}$ in discrete space and thus helps in controlled exploration of the search space. This is illustrated in Part B of Appendix.

The local improvement is expected to produce a better or an equally good solution in terms of the objective functional value $\mathcal{F}$, i.e., $\mathcal{F}(\tilde{\mathbf{S}}^{(i)}) \leq \mathcal{F}(\mathbf{S}^{(i)})$. If the condition holds, $\mathbf{S}^{(i)}$ is replaced by $\tilde{\mathbf{S}}^{(i)}$ in the population. As the local improvement of an solution is independent of other solutions, it can leverage the parallel architecture of the computing platform. The pseudocode of the local improvement operation is provided in Algorithm 2.

4.3 Spatially-aware recombination

During the above phase, the individual solutions are improved locally without any exchange of information between them. Such an approach results in constricted exploration of the search space, and is akin to running a local search algorithm multiple times and reporting the best solution obtained. Interestingly, any two solutions have varying spatial configurations of the zones, and a better solution may be obtained if we can mix the features of individual solutions. Population-based methods permit such recombination of solutions resulting in a wider exploration of the search space. However, the linear nature of the move is not suited to spatially-constrained search spaces. To circumvent the issue, we propose the spatially-aware recombination as outlined in Algorithm 3.

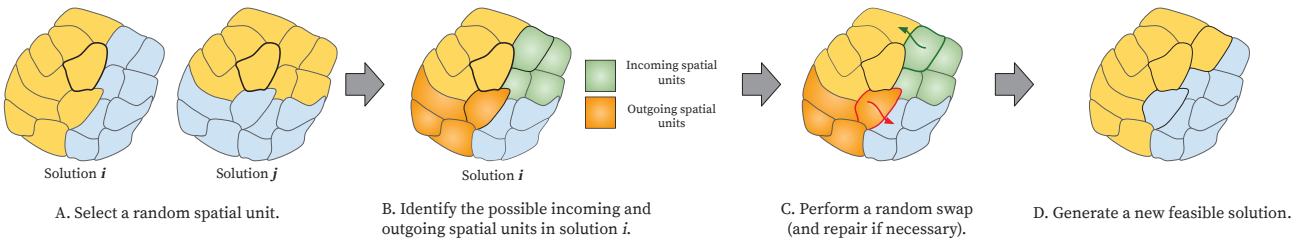


Figure 3: Illustrating the individual steps involved in the spatially-aware recombination operator.

Algorithm 2: Local improvement

Input : $\mathcal{X}$: Population of solutions, $\mathcal{G}$: Contiguity graph
Output : Updated solution
begin
  for $i = \{1, 2, \ldots, SN\}$ **do**
    $\tilde{\mathrm{S}}^{(i)}, \mathcal{I}^{(i)}$: Make a copy of the i^{th} solution and find its indexset
    z: Randomly pick a zone from $\{1, 2, \ldots, K\}$
    // Find nodes adjacent to (but not present in) zone z
    $\mathcal{N}(z) = \left\{v | v \in \mathcal{V} \setminus \mathcal{I}_z^{(i)}, \exists u \in \mathcal{I}_z^{(i)} \text{ and } W_{v,u} = 1\right\}$
    v : From $\mathcal{N}(z)$ pick a random node $v \in \mathcal{I}_w^{(i)}, w \neq z$
    // Move node v from zone w to z and update assignment
    $\tilde{S}_{zv}^{(i)} \leftarrow 1, \tilde{S}_{zw}^{(i)} \leftarrow 0, \quad \mathcal{I}_z^{(i)} \leftarrow \mathcal{I}_z^{(i)} \bigcup \{v\}, \mathcal{I}_w^{(i)} \leftarrow \mathcal{I}_w^{(i)} \setminus \{v\}$
    // Repair zones z and w if they are non-contiguous
    for *zone* $r \in \{z, w\}$ **do**
      $\mathbb{C}_r$: Find the number of connected components in zone r
      if $|\mathbb{C}_r| > 1$ **then**
        // zone r is disconnected and needs repairing
        for $C \in \mathbb{C}_r$ **do**
          if $\exists v \in C | v \in \overline{\mathcal{V}}$ **then**
            continue ▷ contains a seed node
          else
            Assign node $v, \forall v \in C$ to neighboring zones and update the assignment matrix $\tilde{\mathrm{S}}^{(i)}$
    if $\mathcal{F}\left(\tilde{\mathrm{S}}^{(i)}\right) \leq \mathcal{F}\left(\mathrm{S}^{(i)}\right)$ **then**
      $\mathrm{S}^{(i)} \leftarrow \tilde{\mathrm{S}}^{(i)}$ ▷ Fitness-based selection

Algorithm 3: Spatially-aware recombination

Input : $\mathcal{X}$: Population of solutions, $\mathcal{G}$: Contiguity graph
Output : Updated solution
begin
  $\mathrm{F} = \left\{ \frac{1}{1+|\mathcal{F}\left(\mathrm{S}^{(i)}\right)|} \middle| \forall i = 1, 2, \ldots, SN \right\}$ ▷ Fitness values
  for $i = \{1, 2, \ldots, SN\}$ **do**
    $\tilde{\mathrm{S}}^{(i)}, \mathcal{I}^{(i)}$: Make a copy of the i^{th} solution
    Based on the fitness value, probabilistically select the j^{th} solution, i.e., $\mathrm{S}^{(j)}$, such that $j \in \{1, 2, \ldots, SN\} \setminus \{i\}$
    $\mathcal{I}^{(i)}, \mathcal{I}^{(j)}$: Determine the indexset of the i^{th} and the j^{th} solution
    Randomly pick a node $u, u \in \{1, 2, \ldots, N\}$, so that there is a zone $z, \exists z \in \{1, 2, \ldots, K\} | u \in \mathcal{I}_z^{(i)} \cap \mathcal{I}_z^{(j)}$
    // Find two sets of nodes for doing simultaneous swap
    $I_z = \left\{v | v \in \mathcal{I}_z^{(j)} \setminus \mathcal{I}_z^{(i)}\right\}, O_z = \left\{u | u \in \mathcal{I}_z^{(i)} \setminus \mathcal{I}_z^{(j)}\right\}$
    Randomly pick an incoming node $v \in I_z$ and an outgoing node $u \in O_z$
    In the i^{th} solution, simultaneously insert node v into zone z and remove node u from zone z, and update the assignments in $\tilde{\mathrm{S}}^{(i)}$
    Repair the zones in the i^{th} solution that were rendered non-contiguous by the swap operation
  // Update the population synchronously based on fitness
  for $i = \{1, 2, \ldots, SN\}$ **do**
    if $\mathcal{F}\left(\tilde{\mathrm{S}}^{(i)}\right) \leq \mathcal{F}\left(\mathrm{S}^{(i)}\right)$ **then**
      $\mathrm{S}^{(i)} \leftarrow \tilde{\mathrm{S}}^{(i)}$

The recombination operator works with two solutions, say i and j, with the latter selected probabistically based on the fitness value. The fitness function is defined to allow solutions with lower functional value have higher fitness as this is a minimization problem[4]. Next, a solution is modified by swapping spatial units in the following steps. Suppose a zone z is present in both solution i and j as $z^{(i)}$ and $z^{(j)}$ such that they have a common node. Now, $z^{(i)}$ is modified by simultaneously inserting a node v (present in $z^{(j)}$) into it while removing a node u (present in $z^{(i)}$) from it. The operator is designed to steer solution i towards the fitter solution j by exchanging information. In doing so, we expect to find intermediate solutions that may have better fitness than the incumbent solution.

Note that the swapping of spatial units might lead to presence of holes in the involved zones and may break contiguity. To avoid such undesirable situations, we prefer to use boundary units[5] while performing the swaps. Nevertheless, the repair operation still needs to be applied if the contiguity of the zones are broken by the swap.

[4]For maximization, the fitness can be set equal to the objective functional value.
[5]Spatial units in zone z that share boundary with other zones.

The recombination steps are illustrated in Figure 3. The newly generated solutions are updated in the population synchronously, i.e., a new set of SN solutions are generated before performing the one-to-one fitness-based replacement (selection) of the solutions.

4.4 Putting it all together

The full pseudocode of SPATIAL is in Algorithm 4. The code is available at https://github.com/subhodipbiswas/SPATIAL.

Algorithm 4: SPATIAL

Data: $\mathcal{G}$: Contiguity graph, $\mathcal{F}$: Objective function, Design constraints, Algorithm parameters
Result: Best spatial configuration
begin
  $\mathcal{X}$: Initialize the population of solutions using Algorithm 1.
  for $iter = 1, 2, \ldots, iter_{\max}$ **do**
    Modify $\mathcal{X}$ through local improvement (Algorithm 2)
    Update $\mathcal{X}$ by spatial recombination (Algorithm 3)
  X^*: Find the best solution in $\mathcal{X}$
return : X^*

Several remarks are in order. *Firstly*, the search moves (local search and recombination) are performed based on the adjacency relationships between the spatial units. This is analogous to the ABC framework where honeybees search the neighborhood of a food source for finding a better one. *Secondly*, the search moves do not require beforehand information about the gradient or put any restriction on the objective/constraint function(s), such as linearity; thus can be used as a black-box optimizer for solving any SOPs. *Thirdly*, we do not prevent a move that breaks contiguity. The solution is repaired and in the process leads to (local) exploration of the search space. *Lastly*, the local search and the recombination (synchronous population update) operations can make use of parallel architecture thereby resulting in a significant reduction in computation time without affecting the solution quality.

5 EXPERIMENTATION

In this section, we evaluate our technique against traditional baseline methods on the problem of school boundary formulation. The dataset description, baseline methods, experimental setting, performance metrics, and result discussions are provided subsequently. Additionally, we perform detailed ablation tests to understand the efficacy of the individual components in our proposed framework.

5.1 Dataset and pre-processing

For this study, we selected two school districts (corresponding to counties) located in the eastern part of the USA. These large-sized districts are characterized by widely varying demographics and recent population growth, thereby making the problem instances challenging for spatial algorithms. We used GIS data of both the districts for the school year 2019-2020. It contains information about

- **SPAs**: Location coordinates of the boundary and aggregated student count at different school levels.
- **Schools**: Location coordinates of the school building, school level and its program capacity.

The summary statistics of the datasets are provided in Table 1. From the given data, we determined the adjacency relationship between the nodes (spatial units) and the center nodes (units that contain schools) using Python-based geospatial packages like PySal [33].

Table 1: Summary statistics of the school districts

District	# SPAs (N)	# Schools (K)		
		Elementary	Middle	High
X	453	57	16	16
Y	1313	138	26	24

5.2 Baseline methods

We evaluate our SPATIAL method against the following baselines:

- **Basic Hill Climbing (BHC)** [4]: A variation of Hill Climbing that starts with an initial feasible solution and searches for better neighboring solutions in a random manner. If such a solution is found, it is saved and the search continues till a local optimum is obtained.
- **OBA (Old Bachelor Acceptance)** [18]: A thresholding algorithm that applies similar search move as BHC, however, the new solutions are accepted if it is better (or worse) within some acceptable thresholds.
- **SA (Simulated Annealing)** [21]: A stochastic version of the BHC that closely follows the process of tempering of metals. It allows for worsening moves to take place if no better solutions are found and can escape local optima.
- **TS (Tabu Search)** [11]: Uses a restrictive (tabu) list to forbid revisiting recently explored solutions so that the new neighboring solutions can be explored.

We used the default parametric configuration for each of the baseline methods, as suggested in the literature. For SPATIAL, we set the population size to 10. All the algorithms were made to run for till the change in objective function value was less than 0.01 for 10 consecutive iterations, or the limit of 1000 iterations was reached. **Note:** Initially, we also simulated two more methods – greedy randomized adaptive search procedure (GRASP) [34] and Mixed Integer Linear Programming (MILP) [37]. While GRASP's performance was inferior to the other baselines, MILP could not converge to an optimal solution (even with a run-time of 24 hours) for 4/6 test cases even. Hence, these methods are not included in the reported results for comparison.

5.3 Objective criteria and Evaluation metrics

We simulated the peer algorithms to solve the school boundary formation problem for each problems instance as reported in Table 1. The weighing parameter λ is empirically set to 0.7 and 0.8 for districts **X** and **Y** respectively so that the condition $\lambda/1-\lambda \geq 2$ is satisfied.

We used two metrics for assessing the quality of solutions produced by the algorithms. Both the metrics can be interpreted as percentage scores since they lie in the range $[0, 100]$.

- **Balance**: This metric measures the average balance between a school's program capacity and the number of students residing within its boundary. It is calculated as

$$\frac{1}{K}\sum_{z=1}^{K}\left(100\times\left|1-\left|1-\frac{\sum_{v=1}^{N} S_{zv}\cdot {}_{\mathrm{L}}n_v}{\sum_{v=1}^{N} S_{zv}\cdot {}_{\mathrm{L}}c_v}\right|\right|\right). \tag{4}$$

  We penalized both under-enrolled and overburdened schools equally with respect to the capacity of schools. This is an important metric for school planners since most of the boundary changes occur to achieve a better balance in schools.
- **Compactness**: This metric measures how tighly a school's boundary is packed on an average with respect to its perimeter. A scaled version of the Polsby-Popper metric [32] is used to measure compactness as

$$\frac{1}{K}\sum_{z=1}^{K}\left(100\times\frac{4\pi\cdot\mathrm{Area}(\mathcal{I}_z)}{\mathrm{Perimeter}^2(\mathcal{I}_z)}\right). \tag{5}$$

  Compactness has a direct bearing on a budget of a school district. Compact school boundaries indirectly translate to proximal schools that students can walk to and thereby lower the transportation cost incurred by the school district.

Table 2: Performance of peer algorithms on the problem of school boundary formation in both the districts.

Models	District X					
	Elementary School		Middle School		High School	
	Balance	Compactness	Balance	Compactness	Balance	Compactness
Present	83.5671 ± 0.0000	32.5344 ± 0.0000	85.7586 ± 0.0000	27.3467 ± 0.0000	87.9101 ± 0.0000	27.3467 ± 0.0000
HC	87.4392 ± 0.8087	36.4436 ± 1.2762	93.0557 ± 2.1933	30.3058 ± 1.6974	96.5058 ± 1.9421	28.0142 ± 2.4857
OBA	86.5387 ± 1.1054	35.0348 ± 1.0757	91.5191 ± 2.7554	29.0308 ± 2.5145	94.3810 ± 1.9450	25.7212 ± 3.2835
SA	86.9885 ± 0.9542	36.2501 ± 1.5118	92.3514 ± 2.3691	29.6381 ± 3.7883	94.9052 ± 1.6857	27.6893 ± 3.4297
TS	87.6545 ± 0.6669	37.3322 ± 1.7913	93.2058 ± 2.9611	31.0033 ± 2.0849	95.3739 ± 1.6518	28.9664 ± 2.8948
SPATIAL	**88.0687 ± 0.4367**	**42.4213 ± 1.3372**	**95.2043 ± 1.4368**	**37.5817 ± 2.3496**	**98.0835 ± 0.7759**	**36.8147 ± 1.9898**
Models	District Y					
	Elementary School		Middle School		High School	
	Balance	Compactness	Balance	Compactness	Balance	Compactness
Present	84.6901 ± 0.0000	**35.9234 ± 0.0000**	84.9227 ± 0.0000	27.7097 ± 0.0000	88.4553 ± 0.0000	26.8001 ± 0.0000
HC	92.5992 ± 0.6034	29.8237 ± 0.7715	90.6346 ± 0.4646	23.7399 ± 2.2729	94.7308 ± 3.6228	22.3509 ± 3.0465
OBA	89.8471 ± 0.7397	27.1554 ± 0.8889	89.6780 ± 0.7126	18.5098 ± 1.8385	91.3610 ± 1.6740	18.0583 ± 1.5202
SA	92.5673 ± 0.5876	29.7977 ± 1.2676	**90.7170 ± 0.3586**	24.1184 ± 2.5154	95.8883 ± 1.3153	21.4539 ± 3.0878
TS	92.4823 ± 0.3777	31.2161 ± 0.9821	90.6917 ± 0.3752	27.0545 ± 2.5537	96.0029 ± 0.8505	25.0559 ± 2.4043
SPATIAL	**92.6013 ± 0.6179**	34.5602 ± 1.143	90.5901 ± 0.3098	**33.6391 ± 2.0858**	**96.5190 ± 0.3046**	**29.5485 ± 1.3143**

5.4 Discussions and Results

We observed that the two objectives- balance and compactness - are conflicting in nature, and thus a spatial optimizer has to make a trade-off between them while selecting a new solution. Interestingly, solving the ES instance of the problem is particularly challenging. This is because a school district that has seen recent population growth is likely to have new ESs, which are situated at arbitrary locations without much separation between them. This leads to a decrease in the clustering tendency. Besides, there is a high variation in the capacity of the ESs with the new ones having much higher capacity than its older counterparts. In attempting to satisfy the schools' capacities, the spatial algorithms might try to fill in concave segments in the SAZs with regular-shaped SPAs having a high density of student population, thereby affecting the balance scores, which are generally lower than its other counterparts. The MS and HS, being well-separated and showing fewer deviations of capacity, are comparatively easier to solve. Interestingly, the ES boundaries are more compact than their MS and HS counterparts. In comparison to ES SAZs, a greater proportion of SAZs of MS and HS share borders with the school district boundary, which is usually zigzagged by naturally occurring geographies (highly irregular geometries). These additional considerations, besides the spatial constraints, make school boundary formation a challenging spatial optimization algorithm to solve.

For comparison purposes, we simulated 25 trial runs of each baseline and record the final solutions. In Table 2, we reported the mean and standard deviation of the performance metrics. We also included the current school boundary configuration of the school districts. It is marked as Present in the table. The results reveal that SPATIAL is able to generate better solutions (spatial configurations) in the majority of the test cases, especially for district **X**. For district **Y**, the performance of our method is comparable to the baselines in terms of the balance scores, yet the compactness scores are much better. SPATIAL is the only model that can achieve at par or better compactness than the existing configuration of district **Y**. A similar trend has also been observed for district **X**. The baseline methods adopt a greedy approach by continuing to look for better solutions in the locality of the incumbent solution. In doing so, the solutions lying just outside their immediate neighborhood remain elusive to them. On the other hand, aided by the recombination operator and the repair mechanism, SPATIAL is able to find these solutions while maintaining diversity. This leads to an overall better performance on both the metrics.

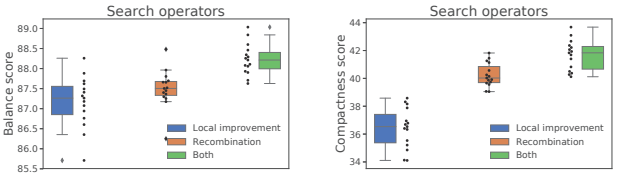
5.5 Ablation study of SPATIAL

In this subsection, we conduct ablation studies to understand the effectiveness of different components in the proposed SPATIAL framework. To this end, we perform the simulations on two instances (ES and MS) of the district **X**. We omitted HS instance since it is similar to MS in both problem complexity and difficulty.

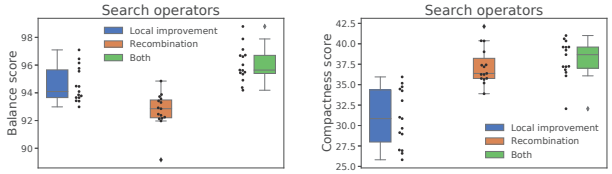
How effective are the search operators in improving solution quality? To answer this question, we simulate 15 sample runs of SPATIAL with three possible configurations as stated below:

- Only the local improvement operator is activated.
- Only the recombination operator is activated.
- Both the operators are activated.

The performance metrics obtained by each configuration are illustrated as error bars in Figure 4. We observe that using the local improvement operator does not guarantee the high-performance gain. It also shows a high standard deviation indicating the effect of initialization on output performance. On the other hand, the recombination operation is found to particularly aid in achieving high compactness. As mentioned earlier, recombination helps to find better solutions lying outside the immediate neighborhood of the incumbent solution. To sum up, we noticed that the combined effect of both the operators led to an overall improvement in solution quality.



(a) Elementary school



(b) Middle school

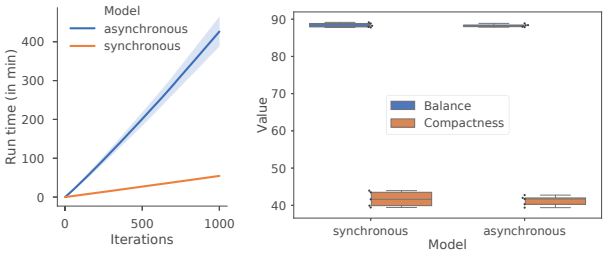
Figure 4: The performance metrics obtained by the different combinations of search operators in SPATIAL . We observed that the combined effect of both the operators resulted in better quality solutions.

Should the population update be made synchronously or asynchronously? The fitness-based selection to update the solutions during the recombination operation can be performed in two ways as follows:

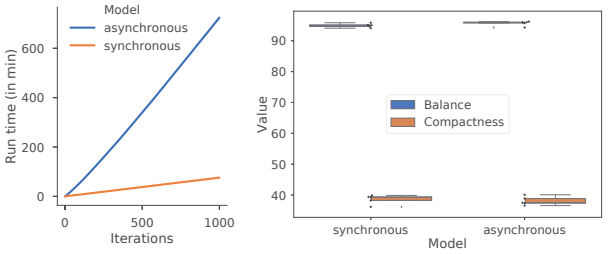
- **Asynchronous update**: The population of (parent) solutions is scanned sequentially, and as soon as a new (offspring) solution is generated, the parent is replaced if the offspring is found to be better or equally good.
- **Synchronous update**: The offspring solutions are maintained separately as they are generated via recombination. Post generation, a one-on-one comparison is made between the offspring and its corresponding parent, and the better one is retained in the population.

Asynchronous update follows 'serial' execution to update the population of solutions. As the (fitter) offspring solutions are updated in the population, they become available to the remaining parent solutions that are yet to be updated. This makes the algorithm more greedy and is prone to reaching local optima faster. On the other hand, asynchronous update prevents intermixing of parent and offspring solutions by delaying the update. This has two-fold advantage: (a) helps to preserve population diversity, and (b) enables usage of 'parallel' architecture, i.e., multiple copies of the population are sent to the individual cores with each generating one offspring, leading to computational efficiency.

To observe the difference in performance between them, we simulated 5 runs of each update on a 16-core machine with 2.2 GHz Intel CPUs having x86_64 architecture and running on Ubuntu 16.04.6 LTS operating system. The obtained run-times and performance metrics are plotted in Figure 5 for two instances of District **X**. Interestingly, we observed no marked difference in performance with a noticeable gap in their run-times. Hence, we conclude that synchronous update is preferable in the interest of practical time limits.



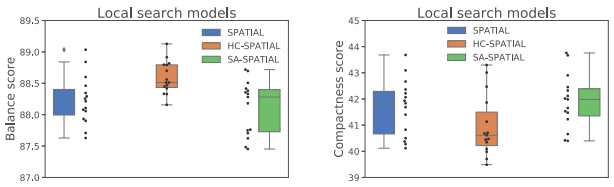
(a) Elementary school



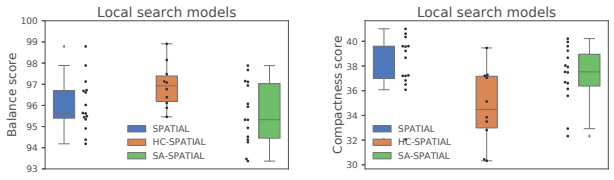
(b) Middle school

Figure 5: The trade-off between run time and performance metrics obtained on district X by asynchronous and synchronous population update techniques. No major difference in performance is observed while the run-time of asynchronous update is significantly higher than its counterpart.

How does SPATIAL fare when other local search methods are integrated? Any local search can easily be integrated within SPATIAL. We instantiated two variations of SPATIAL by integrating the basic Hill Climbing and Simulated Annealing in the local improvement phase, thereby resulting in HC-SPATIAL and SA-SPATIAL, respectively. These two variants are compared with the original SPATIAL in Figure 6. We observed no clear winner.



(a) Elementary school



(b) Middle school

Figure 6: Error plots of performance metrics obtained from 15 sample runs of algorithmic variations of SPATIAL.

6 CONCLUSION & FUTURE WORK

We propose a spatial memetic algorithm SPATIAL in this paper inspired by the field of population-based metaheuristics. To this end, we developed search operators that are spatially cognizant and are able to seek improved solutions by searching through discrete search space characterized by spatial constraints. Our SPATIAL method achieved better performance than traditional baseline algorithms on the problem of school boundary formation. A detailed analysis revealed interesting insights into the effectiveness of the different components– recombination operator and synchronous population update. Moreover, we highlight the generalizability of our framework by integrating different local search techniques. Our contribution lies in showing the potential of memetic algorithms in solving spatial optimization problems.

For future research, we have the following directions in mind. Firstly, we may develop a many-objective version of our algorithm where multiple decision criteria can be considered in solving the problem. Secondly, we can investigate the effect of population size on the recombination search operator. It can also be augmented using MCMC techniques for sampling better solutions. Lastly, SPATIAL can help in getting an initial set of good solutions, which can be used to prefix decision variables so that exact optimization methods can solve mid- to large-sized SOPs in a reasonable time.

ACKNOWLEDGMENTS

This research is supported in part by National Science Foundation grants DGE-1545362 and IIS-1633363. We are grateful to Jessica Gillis, Susan Hembach, Paul Ngo, Pranita Ranbhise and Andreea Sistrunk for providing us with the data and helpful insights regarding the school boundary process. **Disclaimer:** The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any school board, NSF, or the U.S. Government.

REFERENCES

[1] Subhodip Biswas, Fanglan Chen, Andreea Sistrunk, Sathappan Muthiah, Zhiqian Chen, Nathan Self, Chang-Tien Lu, and Naren Ramakrishnan. 2020. Geospatial clustering for balanced and proximal schools. In *AAAI*. 13358–13365.
[2] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. 2011. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing* 11, 6 (2011), 4135–4151.
[3] Eric Bonabeau, Marco Dorigo, Directeur de Recherches Du Fnrs Marco, Guy Theraulaz, Guy Théraulaz, et al. 1999. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford university press.
[4] Edmund K Burke, Graham Kendall, et al. 2005. *Search methodologies*. Springer.
[5] Javier Del Ser, Eneko Osaba, Daniel Molina, Xin-She Yang, Sancho Salcedo-Sanz, David Camacho, Swagatam Das, Ponnuthurai N Suganthan, Carlos A Coello Coello, and Francisco Herrera. 2019. Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation* 48 (2019), 220–250.
[6] Juan Carlos Duque, Raúl Ramos, and Jordi Suriñach. 2007. Supervised regionalization methods: A survey. *International Regional Science Review* 30, 3 (2007), 195–220.
[7] James Evans. 1992. *Optimization algorithms for networks and graphs*. CRC Press.
[8] Kenneth C Gilbert, David D Holmes, and Richard E Rosenthal. 1985. A multiobjective discrete optimization model for land allocation. *Management Science* 31, 12 (1985), 1509–1522.
[9] Fred Glover. 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* 13, 5 (1986), 533–549.
[10] Fred Glover and Jin-Kao Hao. 2011. The case for strategic oscillation. *Annals of Operations Research* 183, 1 (2011), 163–173.
[11] Fred Glover and Manuel Laguna. 1998. Tabu search. In *Handbook of combinatorial optimization*. Springer, 2093–2229.
[12] Fred Glover, Manuel Laguna, and Rafael Martí. 2000. Fundamentals of scatter search and path relinking. *Control and cybernetics* 29, 3 (2000), 653–684.
[13] Fred Glover, Manuel Laguna, and Rafael Marti. 2003. Scatter search and path relinking: Advances and applications. In *Handbook of metaheuristics*. Springer, 1–35.
[14] Fred Glover and Kenneth Sörensen. 2015. Metaheuristics. *Scholarpedia* 10, 4 (2015), 6532.
[15] Trevor S Hale and Christopher R Moberg. 2003. Location science research: a review. *Annals of operations research* 123, 1-4 (2003), 21–35.
[16] Jiawei Han, Micheline Kamber, and Anthony KH Tung. 2001. Spatial clustering methods in data mining. *Geographic data mining and knowledge discovery* (2001), 188–217.
[17] Catherine M Hosage and Michael F Goodchild. 1986. Discrete space location-allocation solutions from genetic algorithms. *Annals of Operations Research* 6, 2 (1986), 35–46.
[18] Te C Hu, Andrew B Kahng, and Chung-Wen Albert Tsao. 1995. Old bachelor acceptance: A new class of non-monotone threshold accepting methods. *ORSA Journal on Computing* 7, 4 (1995), 417–425.
[19] Jörg Kalcsics, Stefan Nickel, and Michael Schröder. 2005. Towards a unified territorial design approach—Applications, algorithms and GIS integration. *Top* 13, 1 (2005), 1–56.
[20] Dervis Karaboga and Bahriye Basturk. 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization* 39, 3 (2007), 459–471.
[21] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.
[22] Philip M Lankford. 1969. Regionalization: theory and alternative algorithms. *Geographical Analysis* 1, 2 (1969), 196–212.
[23] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha-da Gama. 2019. Introduction to location science. In *Location science*. Springer, 1–21.
[24] Arika Ligmann-Zielinska. 2017. *Spatial Optimization*. American Cancer Society, 1–6. https://doi.org/10.1002/9781118786352.wbieg0156
[25] Nimrod Megiddo and Kenneth J Supowit. 1984. On the complexity of some common geometric location problems. *SIAM journal on computing* 13, 1 (1984), 182–196.
[26] Jeremy Mennis and Diansheng Guo. 2009. Spatial data mining and geographic knowledge discovery—An introduction. *Computers, Environment and Urban Systems* 33, 6 (2009), 403–408.
[27] Efrén Mezura-Montes and Carlos A Coello Coello. 2011. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation* 1, 4 (2011), 173–194.
[28] Pablo Moscato et al. 1989. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report* 826 (1989), 1989.
[29] Pablo Moscato and Carlos Cotta. 2010. A modern introduction to memetic algorithms. In *Handbook of metaheuristics*. Springer, 141–183.
[30] Stuart S Nagel. 1964. Simplified bipartisan computer redistricting. *Stan. L. Rev.* 17 (1964), 863.
[31] Stan Openshaw and Liang Rao. 1995. Algorithms for reengineering 1991 Census geography. *Environment and planning A* 27, 3 (1995), 425–446.
[32] Daniel D Polsby and Robert D Popper. 1991. The third criterion: Compactness as a procedural safeguard against partisan gerrymandering. *Yale Law & Policy Review* 9, 2 (1991), 301–353.
[33] Sergio J Rey and Luc Anselin. 2010. PySAL: A Python library of spatial analytical methods. In *Handbook of applied spatial analysis*. Springer, 175–193.
[34] Roger Z Ríos-Mercado and Elena Fernández. 2009. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research* 36, 3 (2009), 755–776.
[35] Aaron J. Saiger. 2010. The School District Boundary Problem. *The Urban Lawyer* 42, 3 (2010), 495–548. http://www.jstor.org/stable/27895808
[36] María Angélica Salazar-Aguilar, Roger Z Ríos-Mercado, and Mauricio Cabrera-Ríos. 2011. New models for commercial territory design. *Networks and Spatial Economics* 11, 3 (2011), 487–507.
[37] Takeshi Shirabe. 2009. Districting modeling with exact contiguity constraints. *Environment and Planning B: Planning and Design* 36, 6 (2009), 1053–1066.
[38] Kenneth Sörensen and Fred Glover. 2013. Metaheuristics. *Encyclopedia of operations research and management science* 62 (2013), 960–970.
[39] Daoqin Tong and Alan T Murray. 2012. Spatial optimization in geography. *Annals of the Association of American Geographers* 102, 6 (2012), 1290–1309.
[40] Ningchuan Xiao. 2008. A unified conceptual framework for geographical optimization using evolutionary algorithms. *Annals of the Association of American Geographers* 98, 4 (2008), 795–817.
[41] Jing Yao, Xiaoxiang Zhang, and Alan T Murray. 2018. Spatial optimization for land-use allocation: accounting for sustainability concerns. *International Regional Science Review* 41, 6 (2018), 579–600.
[42] Andris A Zoltners and Prabhakant Sinha. 1983. Sales territory alignment: A review and model. *Management Science* 29, 11 (1983), 1237–1256.

A ABC ALGORITHM

ABC is a continuous-valued real-parameter global optimization algorithm based on the foraging behavior of honeybees [20]. The pseudo-code of the ABC method for solving a bound-constrained real-parameter optimization problem is provided in Algorithm 5.

Algorithm 5: ABC algorithm

Data: $\mathcal{F}$: Objective function, D : Problem dimensionality, SN : Population size, $iter_{\max}$: Evaluation budget, $limit$: Non-improvement limit
Result: Best possible solution to $\mathcal{F}$
begin
  // Initialize the candidate solutions within the bounds
  for $i = 1, 2, \ldots, SN$ **do**
    for $j = 1, 2, \ldots, D$ **do**
      $x_{i,j} = x_{j,\min} + rand_{i,j}[0,1] \times (x_{j,\max} - x_{j,\min})$
  Evaluate the solutions
  // Iterative improvement of the solutions till termination
  for $iter = 1, 2, \ldots, iter_{\max}$ **do**
    // Employed bee phase
    for $i = 1, 2, \ldots, SN$ **do**
      Pick a random index $k \in \{1, \ldots, SN\} \setminus \{i\}$
      for $j = 1, 2, \ldots, D$ **do**
        $v_{i,j} = x_{i,j} + rand_{i,j}[-1,1] \times (x_{i,j} - x_{k,j})$,
      if $\mathcal{F}(\vec{V}_i) \leq \mathcal{F}(\vec{X}_i)$ **then**
        $\vec{X}_i \leftarrow \vec{V}_i$
        $count_i \leftarrow 0$
      else
        $count_i \leftarrow count_i + 1$
    // Onlooker bee phase
    for $i = 1, 2, \ldots, SN$ **do**
      Repeat the steps of Employed Bee phase except selecting the k^{th} solution based on the probability calculated in Eq. (7)
    // Scout bee phase
    for $i = 1, 2, \ldots, SN$ **do**
      if $count_i \geq limit$ **then**
        Reinitialize $\vec{X}_i$ randomly
        $count_i \leftarrow 0$
  Find the best solution $\vec{X}^*$, i.e., $\mathcal{F}(\vec{X}^*) \leq \mathcal{F}(\vec{X}_i)$, $i = 1, \ldots, SN$
return $:\vec{X}^*$

In ABC terminology, a food source is a metaphor for a candidate solution, i.e., a $D-$dimensional real-parameter vector

$$\vec{X}_i = \left[x_{i,1}, x_{i,2}, \ldots, x_{i,D}\right], \qquad i = 1, \ldots, SN,$$

where SN is the number of food sources, and each parameter may lie within some bounded range of values. The nectar amount of each food source is determined by the value of its objective (fitness) function $f(\vec{X}_i)$. Following the random initialization of the food sources, ABC searches for a global optimum in the $D-$dimensional real-parameter space through the action of artificial bees (search agents). The search for an optimal solution is performed iteratively in three phases till a termination criteria is satisfied.

Employed bee phase. In this phase, the employed bees bring about improvement in the quality of food sources by exploring it's neighborhood. The local exploration is implemented as

$$v_{i,j} = x_{i,j} + \phi_{i,j} \times (x_{i,j} - x_{k,j}), \tag{6}$$

where $k \in \{1, 2, \ldots, SN\} \setminus \{i\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indices, and $\phi_{i,j}$ is another random number lying in the range $[-1, 1]$. $\vec{V}_i = \left[v_{i,1}, v_{i,2}, \ldots, v_{i,D}\right]$ represents the position of a new food source that an artificial bee has found in the vicinity of original food source $\vec{X}_i$. If the new position has better fitness value, i.e., $f(\vec{V}_i) \geq f(\vec{X}_i)$, the artificial bee replaces the position of $\vec{X}_i$ by $\vec{V}_i$ in its memory. This fitness-based (greedy) selection helps in preserving better solutions as the search progresses.

Onlooker bee phase. This is identical to the above phase except for the probabilistic selection of the food source i based on the value

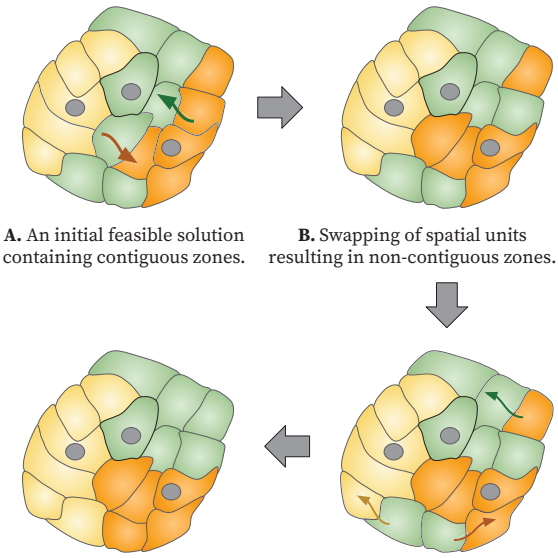
$$p_i = f(\vec{X}_i) / \textstyle\sum_{i'=1}^{SN} f(\vec{X}_{i'}). \tag{7}$$

Scout bee phase. This is random search move for re-initializing food sources which could not be improved by the above two phases for a pre-defined number of iterations.

Note that the linear search move in Equation (6) is adopted by ABC to generate new solutions in real-parameter space. This move is suited to continuous search space and cannot be used to solve discrete optimization problem. Our SPATIAL framework executes the spatial analogue of these linear search moves for solving SOPs.

B REPAIR PROCESS

The repair mechanism is illustrated in Figure 7. It results in a new feasible solution that is a few hops away from the initial solution. We opine that the repair process results in attaining solutions that are otherwise difficult to reach from the initial solution and thus leads to efficient exploration of the search space.



A. An initial feasible solution containing contiguous zones.

B. Swapping of spatial units resulting in non-contiguous zones.

D. A new feasible solution is generated as a result of the repair process.

C. Repair mechanism: Randomly assign the connected components of non-contiguous zones (that does not contain a center node) to neighboring zones.

Figure 7: Illustration of the individual steps involved in applying the repair process once a solution becomes infeasible. The spatial units with gray circles inside them are center nodes that cannot be reassigned.