# Integrating Knowledge Representation Techniques for Land Use Management Optimization

Margaux Van Geem[1], Benno Kruit[1] and Stefan Schlobach[1]

[1]Vrije Universiteit Amsterdam (VU Amsterdam), De Boelelaan 1105, 1081HV, Amsterdam, Netherlands

**Abstract**

The effective allocation of land resources has become imperative to address the growing sustainability challenges of the modern world. In this context, optimization tools have emerged as key players in supporting decision makers through the complexities of land use management. Among them, the CoMOLA tool stands out due to its flexibility and versatility. However, an extensive investigation of recent publications using the tool uncovered a lack of standardized domain interpretation as well as technical flaws in model implementations. This paper aims to improve the development, extensibility and performance of the optimization tool by integrating knowledge representation techniques. Firstly, an ontological classification was presented as a way to address the existing language discrepancies. Its implementation would allow the preprocessing of domain knowledge within CoMOLA and improve interoperability. Secondly, an abstract model design was introduced, along with a set of ready-to-use simplified objective functions. Finally, further streamlining was pursued through the creation of utility functions to process the data extracted from the generated map individuals, and by the exploration of an adjusted version of the new design including a preprocessing layer of algorithmic data. The time and fitness performance of these two new model structures were compared with the implementation of one published research using the CoMOLA tool. The results indicate that the preprocessing layer negatively affects the execution time of the algorithm, but neither of the revised designs compromises the quality of the solutions. Additionally, they greatly reduce the programming load on users without sacrificing the original customizability. Although the accuracy of the test results could be refined, knowledge representation techniques enhance the usability and scalability of the tool, paving the way for applications beyond land use management.

**Keywords**

Land use management, Optimization algorithms, CoMOLA, Knowledge Representation

## 1. Introduction

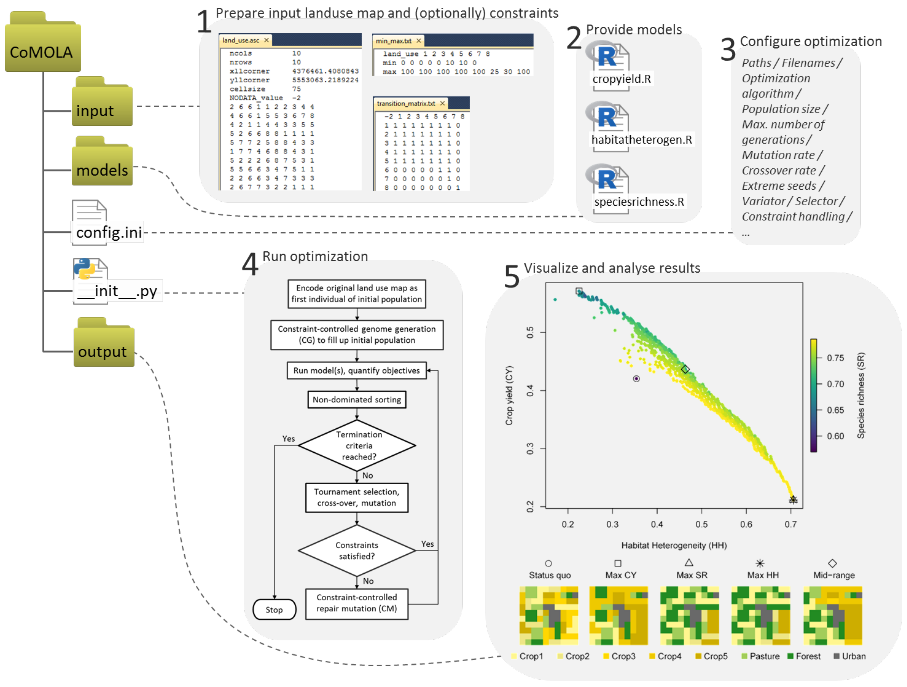
In the contemporary world, the judicious management of resources, particularly land, has become a critical imperative. Policy integration plays a central role in achieving this goal and its implications resonate in environmental sustainability, economic vitality, and societal well-being [1]. However, the continued growth of the global population, the acceleration of urbanization, and the intensification of environmental concerns introduce many complexities that can be challenging to navigate [2, 3, 4].

In response to the increasing demands for effective land use planning, a variety of specialized optimization tools has emerged. These tools, based on mathematical models and computational algorithms, provide decision makers with a platform to explore diverse scenarios with multifaceted objectives [5]. Some, like ArcGIS, which enables users to analyze and visualize spatial data, serve multiple applications, including land use planning [6]. Others are developed to address specific needs. For example, AgriFinland focuses on improving the sustainability of agricultural systems in high-latitude areas such as Finland, while CLUMondo, a land systems-based model, focuses on food and services demand when optimizing changes in land use [7, 8, 9].

### 1.1. Description of CoMOLA

This paper explores the field of land use management optimization, focusing on CoMOLA (Constrained Multi-Objective Land Use Allocation), a multi-objective Python optimization tool designed by Michael

*EKAW 2024: EKAW 2024 Workshops, Tutorials, Posters and Demos, 24th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2024), November 26-28, 2024, Amsterdam, The Netherlands.*
m.vangeem@student.vu.nl (M. V. Geem)


CoMOLA
input
models
config.ini
__init__.py
output
1 Prepare input landuse map and (optionally) constraints
land_use.asc
min_max.txt
transition_matrix.txt
2 Provide models
cropyield.R
habitatheterogen.R
speciesrichness.R
3 Configure optimization
Paths / Filenames / Optimization algorithm / Population size / Max. number of generations / Mutation rate / Crossover rate / Extreme seeds / Variator / Selector / Constraint handling / ...
4 Run optimization
Encode original land use map as first individual of initial population
Constraint-controlled genome generation (CG) to fill up initial population
Run model(s), quantify objectives
Non-dominated sorting
Termination criteria reached?
Yes
No
Tournament selection, cross-over, mutation
Constraints satisfied?
Yes
No
Constraint-controlled repair mutation (CM)
Stop
5 Visualize and analyse results
Crop yield (CY)
Habitat Heterogeneity (HH)
Species richness (SR)
Status quo
Max CY
Max SR
Max HH
Mid−range
Crop1
Crop2
Crop3
Crop4
Crop5
Pasture
Forest
Urban

```
ncols         10
nrows         10
xllcorner     4376461.4080843
yllcorner     5553063.2189224
cellsize      75
NODATA_value  -2
2 6 6 1 1 2 2 3 4 4
4 6 6 1 5 5 3 6 7 8
4 2 1 1 4 4 3 3 5 5
5 2 6 6 8 8 1 1 1 1
5 7 7 2 5 8 8 4 3 3
1 7 7 4 6 8 8 4 3 1
5 2 2 2 6 8 7 5 3 1
5 5 6 6 3 4 7 5 1 1
2 2 6 6 3 4 7 3 3 3
2 6 7 7 3 2 2 1 1 1
```

```
land_use 1 2 3 4 5 6 7 8
min 0 0 0 0 0 10 10 0
max 100 100 100 100 100 25 30 100
```

```
-2 1 2 3 4 5 6 7 8
1 1 1 1 1 1 1 1 0
2 1 1 1 1 1 1 1 0
3 1 1 1 1 1 1 1 0
4 1 1 1 1 1 1 1 0
5 1 1 1 1 1 1 1 0
6 0 0 0 0 0 1 1 0
7 0 0 0 0 0 0 1 0
8 0 0 0 0 0 0 0 1
```

**Figure 1:** Flowchart from the README file of the CoMOLA GitHub repository [10]

Strauch in 2019 to address the intricate challenges of land resource allocation [11]. Figure 1 was sourced from the tool's GitHub page and provides an overview of its constituents and functionalities.

Key components that users are required to provide include an input map, surface and transition constraints, and up to four models in Python or R. The cell values of the input map correspond to one of the ten types of land use defined by the user, ranging from water to urban areas or cropland. The optional constraints mainly allow for more realistic movements of these land use types across the area during the optimization process. For example, considering the increasing need for housing and the elevated transition costs, it is unlikely that an urban area will become forest. Furthermore, the presence of certain natural features, such as an existing watercourse, should not be ignored. This is achieved by specifying the surface ranges for each land use to respect during optimization, proportionate to the total area of the map, or designating certain land uses to be excluded from the optimization process. Finally, the models perform calculations to evaluate the generated map individuals in the form of land use management objective functions such as biodiversity maximization or pollution minimization. Users have complete flexibility in defining the calculations and designing these models from scratch.

Finally, the tool employs the Non-dominated Sorting Genetic Algorithm, utilizing configurations such as population size, generations, mutation rate, etc. During the optimization process, it produces a set of Pareto-optimal solutions, each represented as a land use map alongside its corresponding fitness values. NSGA-II stands out as a well-established and rapid sorting algorithm in elite multi-objective genetic optimization. It excels in simultaneously optimizing each objective without succumbing to dominance by any other solution, setting it apart from single-objective optimization techniques [12].

In summary, CoMOLA offers a versatile platform specialized in optimizing land use management, enabling users to tailor inputs, constraints, and models to their specific needs while utilizing the power of the NSGA-II genetic algorithm for effective multi-objective optimization. Despite its relatively recent introduction to the research community, the adoption of the tool by various authors underscores its relevance in current research trends. its relevance in current research trends. For these reasons, it was chosen as the target optimization tool in this study.

**Table 1**
Comparison of split between map-extracted data and external data of same objective functions across different studies [13, 11, 14]

| Objective Function | Map data | External data |
|---|---|---|
| Maximize economic benefits | Area of commercial use parcels | None |
| Maximize economic benefits | Identification of parcels dedicated to agricultural production | Production intensity and soil fertility |
| Maximize biodiversity | Area of green patches and Shape | Intercept and Slope |
| Maximize biodiversity | Area of forest parcels | Intercept and Slope |

## 1.2. Algorithm Analysis

A first step in this research was to deep dive into the algorithm, to understand how it functions, what kinds of knowledge are required from the user, and how such knowledge is translated into the algorithm implementation, specifically in the context of model design.

### 1.2.1. Optimization Process

In Figure 1, one can observe the overall optimization process of CoMOLA. Since this research focuses on the model design, it is important to understand how they are integrated within the tool. The second point in the flowchart depicts the content of the "models" folder, which contains all the user-provided models in R or Python format. As presented in the file structure blow, which mirrors the example provided in the flowchart, each model is stored in a different subfolder, along with an ASCII file that contains a copy of the land use map to evaluate and a CSV file where the fitness score of that model is stored.

When a new population is generated, each individual is individually evaluated by all the different objectives. To do so, the model main folder is copied as many times as there are individuals in the population, and each individual is duplicated in as many ASCII files as there are models to assess it. In a population, the evaluation of each individual is done sequentially, but the assessment of a single individual across multiple objective functions is conducted simultaneously by running the models on different CPU cores.

```
models/
├── cropyield/
│   ├── cropyield.R
│   ├── map.asc
│   └── cropyield_output.csv
├── habitatheterogen/
│   ├── habitatheterogen.R
│   ├── map.asc
│   └── habitatheterogen_output.csv
└── speciesrichness/
    ├── speciesrichness.R
    ├── map.asc
    └── speciesrichness_output.csv
```

### 1.2.2. Different kinds of knowledge

The term "knowledge" here refers to the expertise and understanding required to design, implement, and configure the optimization tool. In this section a more detailed explanation of the various types of knowledge required and how they manifest in the code is developed. First of all, despite the presence of rather clear documentation, one needs proficiency in Python and R programming, specifically in the implemented library inspyred, from which CoMOLA's optimization algorithm is built around. This

**Table 2**
Comparison of interpretations and calculations of same objective functions across different studies [13, 11, 14]

| Objective Function | Interpretation | Calculation |
|---|---|---|
| Maximize economic benefits | Commercial benefits | Maximize the area of commercial land uses |
| Maximize economic benefits | Crop yield | Function of production intensity level P and soil fertility F |
| Maximize biodiversity | Species richness of vascular plants | Empirical data analysis, General linear model with quasi-poisson distribution of errors |
| Maximize biodiversity | Forest species | Empirical relationship between habitat area and species richness. Function of forest area, a y-intercept constant, and a slope of the species-area-relationship in log-log space |

knowledge is crucial to comprehend the intricacies of the optimization process but also to implement computationally performant models from scratch. Second, knowledge of the tool's configuration settings is essential as it can significantly impact the outcome of experiments. These settings may be specific to the inspyred package, the genetic algorithm utilized (e.g., number of generations, crossover rate, mutation rate), or domain knowledge constraints. Finally, extensive knowledge of the optimization problem at hand, often referred to as domain knowledge, is necessary. This knowledge is evident in the inputs, constraints, and models and plays a vital role in shaping the optimization approach.

### 1.2.3. Model implementations

The analysis of the algorithm's code shows that, out of all the necessary kinds of knowledge required, the implementation of domain knowledge seems the most demanding from a user perspective. Therefore, it can be inferred that the target users of the tool are experts in land use management and related environmental sciences. These users may not necessarily possess extensive programming knowledge. While CoMOLA emphasizes its user-friendliness and flexibility, an examination of how the models are integrated in various research studies using the tool revealed shortcomings from a technical perspective. To conduct this investigation, multiple authors were contacted and graciously provided their code implementations, allowing insightful observations.

Throughout the reviewed implementations, a distinct pattern emerges where there is a clear separation between the processing of data extracted from the generated individual maps and external data. The map-extracted data typically involves algorithmic calculations and manipulations performed within the CoMOLA tool on matrix-related attributes such as cell size, edge, or value, here represented by land use indexes. However, external data refer to additional domain knowledge that cannot be inferred solely from the generated land use maps, which is integrated to enhance the tool's capacity to model real-world scenarios accurately. Examples of this division can be observed in Table 1.

Moreover, the absence of a model design guidelines format raises concerns about code efficiency due to several factors. Firstly, all map and external data processing is confined within each individual model file, potentially leading to suboptimal time and space complexity and under-utilization of available libraries. This approach may also result in redundant code, especially with regard to map data processing, where the same function is repeated across multiple models. Additionally, there is a lack of generalizability, as evidenced by hard-coded file paths within the code. An example of the implementation of the model extracted from the research can be seen on the left side of Figure 9, illustrating some of the points mentioned above.

## 1.3. Literature Review

In addition to the structural analysis of model implementations within studies integrating the CoMOLA tool, five studies were further reviewed to uncover similarities and dissimilarities in the domain analysis of their model implementations.

This investigation revealed a very diverse landscape of implementations, characterized by a multitude of interpretations and calculations of land use management functions. In fact, each author seemed to tailor the tool and, more specifically, the model designs, to their specific research context, introducing unique nuances and variations. For instance, different authors, even when addressing similar objectives, often presented distinctive perspectives and considerations influenced by the specific characteristics of the land use context under investigation. This lack of consensus on the interpretation of land use management functions was highlighted in Table 2, in which various interpretations and calculations of the maximization of economic benefits and biodiversity were reported in the reviewed research. Moreover, it is widely acknowledged that this lack of consensus is not limited to objective functions, but also extends to key domain concepts, such as biodiversity [15].

Furthermore, it should be mentioned that other common aspects were identified in the reviewed studies. Firstly, researchers often opt for simplified input maps and land use management functions compared to real-life scenarios, citing reasons such as computational limitations [14, 16, 11, 17]. Secondly, several studies raise concerns about the tool's time performance, which decreases with the integration of more complex models, larger population sizes, more generations, and larger individual sizes, causing the number of possible combinations to grow exponentially [13, 14].

### 1.4. Research Question

This study contributes to the wider discourse on the intersection of technology and land use management, where optimization tools serve as a vanguard in navigating the complexities of our evolving landscapes. The extensive analysis of the CoMOLA land use management optimization tool has revealed aspects that could potentially benefit from the integration of knowledge representation techniques. In fact, these have shown their efficacy in improving the diversity and quality of optimizations' optimal solutions in real-world applications [18, 19]. Consequently, the pivotal question driving this research is:

- How does the incorporation of knowledge representation techniques for model design impact the development, extensibility, and performance of land use management optimization algorithms?

Parallel to this primary inquiry, the study will also address the following:

- How can common functionalities of such models be separated to increase their robustness?
- How does the integration of model formatting improve the tool's user-friendliness?

## 2. Methods

### 2.1. Ontological Language Unification

Ontologies, within the realm of information science, have proven to be a valuable initial step in the development of resilient knowledge-based systems [20]. In the context of land use management, characterized by diverse terminologies and conceptualizations, the development of an ontology has the potential to address existing language discrepancies, improve interoperability, and contribute to a more cohesive understanding of the domain. Unfortunately, an extensive exploration of existing ontologies related to land use management reveals a fragmented landscape. There is no standardized and widely accepted ontological framework for land use management. Similarly to the interpretation of objective functions, the developed ontologies are often tailored to specific research contexts, impeding their reusability in various studies and scenarios [21, 22, 23].

To address this challenge, a new ontological classification was developed, detailed in Figure 6 of the Appendix. It is important to mention that there is no claim of expertise in the domain or of superiority over existing domain knowledge. The aim is to propose a framework that captures essential concepts, relationships, and classifications relevant to land use management optimization tools such as CoMOLA. The first set of classes and relationships was taken from research [24], making the link between two datasets, the Land-Based Classification Standards [25] and the National Land Use Database [26]. Up to this point, an object of the parcel class is defined by a combination of properties such as structure, activity, function, ownership, and size. This class, central to ontology, was then extended with the Eurostat LUCAS classification of land uses and land covers [27]. Now, an object of the parcel class is also defined by either land covers or land uses, or a combination of both. Drawing an analogy with CoMOLA, parcels could be likened to what has been previously described as land use types, identifying each cell of the input and generated maps. This could already support the domain in identifying the parcel composition of their area of interest, based on the observed land covers, land uses, or a combination of both. The last extension was introduced by the creation of a new class containing land use management functions. We consider each function to be aiming to perform calculations on a combination of land uses, land covers, or both. In doing so, we provide a link between the parcel types and the objective functions. For example, maximizing economic benefits can be translated across different levels as follows:

1. Land Use Management Function: Maximize economic benefits
2. Land Use/Land Cover: Maximize commercial or agricultural production land uses
3. Parcel: Maximize parcels that have commercial or agricultural production land uses.

Furthermore, in a hypothetical implementation within the optimization tool, this ontological design could serve several purposes in preprocessing domain knowledge. Based on the user-defined parcel types in the composition of the input map, CoMOLA could suggest relevant land use management functions for the study area. Based on user-defined parcel types and a selected set of land use management functions, CoMOLA could recognize which parcel types should be considered in each objective function model, eliminating the need for user-defined specifications within the models. Overall, adopting a standardized ontological framework could streamline information exchange and lay the foundation for more generalized and scalable land use management optimization models within CoMOLA.

## 2.2. Model Design using Abstraction

The algorithm analysis and literature review revealed flaws in the scalability, maintainability, and robustness of the objective functions implementations. Furthermore, although there seems to be a consistent structure in terms of separation of map data processing and external data processing, CoMOLA does not provide user-friendly or computationally efficient formatting for user-provided models. In the context of computer science, the abstraction of knowledge representation is crucial for developing systems that can efficiently handle and manipulate information [28]. Abstraction refers to the process of simplifying complex concepts by creating a higher-level representation that captures essential characteristics while omitting unnecessary details. Therefore, abstraction appears to be a relevant technique for addressing model integration issues within CoMOLA.

To simplify the model design and ease the programming load on users, the first step was to introduce utility functions. These deal with file manipulation mainly to prevent hardcoded file paths and efficiently process algorithmic data by using established libraries such as NumPy, as outlined below:

1. Count of cells with specified index values
2. Area calculation based on cell count and cell size from the configuration file
3. Identification of all unique pairs of neighbouring cells. The vertical, horizontal (and diagonal) edges are taken into account once. The definition of a neighbor (four or eight cells) is given in the configuration file.
4. Calculation of the perimeter of the map occupied by a target land use by counting cell pairs that contain one cell of the target land use.

The implementation of these helper functions allows domain experts to focus more on integrating external data into their models.

The second step was to formulate a common structure for all objective functions, which led to the creation of the Land Use Optimization Model abstract class, a code snippet of which can be seen in Figure 7 of the Appendix. The structure is outlined as follows:

1. An input matrix is provided as well as a set of parcel classes to filter the region of interest. An unspecified array translates into all the parcel types considered for that optimization model.
2. From the region of interest, specific algorithmic data is extracted. This can vary from cell count to neighboring cell identification, calculated by the utility functions.
3. Calculations are performed on the algorithmic data, optionally integrating external data. An unspecified supplementary matrix means that it does not require external data.
4. Finally, an output file is updated with the resulting fitness score.

This abstract class served as the foundation for creating a suite of generic and simplified objective functions. These functions include maximization of economic benefits, neighboring compatibility, spatial compactness, spatial heterogeneity, and biodiversity. An example of such model, the compatibility model, is illustrated in Figure 8 of the Appendix. By utilizing this framework, users are equipped with a pre-built collection of objective functions, streamlining the process. In the actual model scripts, users would only need to specify the required inputs, integrate external data to enhance the tool's ability to model real-world scenarios more accurately, and invoke the desired helper model class. The comparison between the original implementation of the compatibility model script with the new one is depicted in Figure 9 of the Appendix. Not only this shows a considerable decrease of lines of code for the user to write, with half of the remaining lines dedicated to importing libraries, but this also validates the models' suitability for use in diverse study areas, as the geospatial specification is primarily represented by the set of user-defined land use types. The design of the models is also highly adaptable for increased complexity and customization. For instance, users can extend the implementations by incorporating new custom functions tailored to the specific optimization problem at hand. This flexibility enables users to fine-tune the models to address unique and specific requirements, enhancing the tool's versatility.

Finally, the integration of utility functions and abstract model design would still include the processing of map-extracted data within each model. As a matter of fact, if a user were to implement multiple objective functions that use the same map-extracted data, the same algorithmic data processing would be performed redundantly, likely leading to computational inefficiency. This is why a more streamlined alternative was explored, in the form of a preprocessing layer, in which neighboring cell pairs would be determined as soon as a new individual map is generated. This cell combination matrix would be then used as input for the models rather than the land use map matrix itself. The selection of the function to perform as part of the pre-processing layer was determined on the basis of the study on which the new model designs were tested.

## 2.3. Experiment Design

In order to understand how different model designs influence the performance of the CoMOLA optimization process, different experiments were performed using as reference the research paper titled "Sustainable Multi-objective Optimisation in Land-use Planning based on Non-dominated Sorting Genetic Algorithm (NSGA-II): a Case Study in Alexandria, Egypt" [13]. The author kindly provided their code composed of the inputs, configurations, and models. This allowed for effective testing of a unique experimental configuration for a meaningful comparison of the different model designs.

Therefore, in this research, a fragment of an Egyptian city map is taken as input, as seen in Figure 2. On this map, one can find the following eight parcel types: Arid, Recreational, Mixed Residential - Commercial, Medical, Religious, Educational, Commercial, and Public Amenities. Given the surface and transition constraints provided in Table 3, the objective is to maximize spatial compactness, compatibility, and economic benefits. Each function is defined as follows in the source study:

- **Spatial compactness**: maximizing the number of land parcels in each land-use clusters. In other words, maximize the number of neighboring cells with the same land use.
- **Compatibility**: compatibility signifies the ability of various land use types within an area to coexist without causing adverse or undesirable impacts on each other. The user provides a compatibility matrix as an external source, where each pair of land use types is mapped to a

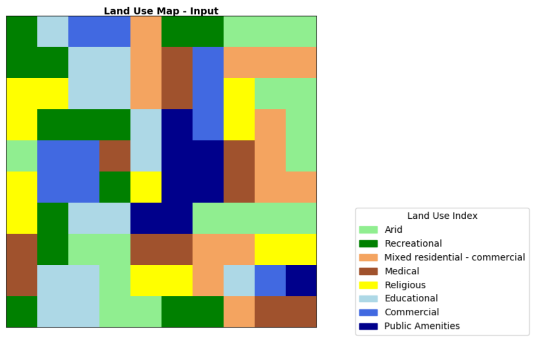


**Figure 2:** Input Map

**Table 3**
Applied surface and transition constraints

| Land Use | Min. Surface (%) | Max. Surface (%) | Allowed Transitions |
|---|---|---|---|
| Arid | 0 | 100 | Any (except P.A.) |
| Recreational | 0 | 100 | Any (except P.A.) |
| Mixed residential - commercial | 0 | 100 | Any (except P.A.) |
| Medical | 0 | 100 | Any (except P.A.) |
| Religious | 0 | 100 | Any (except P.A.) |
| Educational | 0 | 25 | Self and Commercial |
| Commercial | 0 | 30 | Self |
| Public Amenities (P.A.) | 0 | 100 | Self |

compatibility index. The function aims to maximize the compatibility score of all neighboring cells.

- **Economic Benefits**: maximizing the area of the map dedicated to land uses that increase economic benefits. In the implemented scenario, those are specified as Mixed residential - commercial and Commercial.

Three different versions of each objective function were implemented. It is important to mention that the file structure was maintained as described earlier in the various model implementation types. Consequently, each model is composed of a folder which contains a copy of the individual being evaluated in ASCII format, a CSV output file to store the resulting fitness score, and a Python script performing the corresponding model calculations. For each design technique, the models are described as follows:

### 2.3.1. Original Model Design

In the original model design, all the data is processed within the model files. Thus, each model follows the following structure:

1. The first step is to load the individual map and convert it to a NumPy array format.
2. The second step is to extract the data from the input matrix. For the compactness and compatibility models, this translates into the identification of the neighboring region of each cell. On the one hand, compactness further evaluates each region by calculating the amount of same land use type cells within that region. On the other hand, compatibility encompasses additional processing of each region in which the sum of the compatibility scores for each neighboring cell pair. It is relevant to note that for both of these models, the map data processing contains nested loops, and the same pair of cells will be taken into account multiple times. For the economy model, this

translates into searching through the map and counting all the cells with an economy-enhancing parcel type, respectively, Mixed residential - Commercial and Commercial.

3. The third step is optional and consists of integrating additional user-given data. For the compatibility data, the compatibility matrix is loaded and used to calculate the compatibility score of each pair of neighbors processed via the previous step.
4. The fourth step is to print the resulting score.
5. The final step is to open the output file and store the resulting fitness score.

### 2.3.2. Revised Model Design Without Layer

In the revised models without a pre-processing design, the structure from the abstracted class, presented Figure 7 of the Appendix, is implemented as follows:

1. The first step is to load the input and output files, with the input file containing the individual map. For the compatibility model, this is accompanied by the loading of the compatibility matrix file as an external data source. The other models do not have external data defined.
2. The second step is to define the set of parcel types that will be considered by the corresponding objective function. For compactness and compatibility, there are undefined as the entirety of the map is considered. However, for the economy model, an array is provided with the corresponding indexes of Mixed residential - Commercial and Commercial land uses.
3. The third step is to initiate the desired land management helper model with the inputs previously defined. A detailed explanation of each land use management model is provided below:
   a) First, the map data is processed. For compactness and compatibility, the model first uses a utility function to retrieve all unique neighboring cells. For the economy model, a cell count helper function is employed, which counts the cells with an index belonging to the array of land uses provided.
   b) Then, the extracted map data is used to perform calculations. This is only required for compactness and compatibility. For compactness, the pairs composed of the same land use-type cells are counted, whereas for compatibility, the sum of each pair's compatibility score is determined.
   c) Finally, the output file is updated with the resulting score. The printing step present in the original code was skipped in the revised model designs to seek performance improvement.

### 2.3.3. Revised Model Design With Layer

The revised model design with a pre-processing layer is structurally very similar to the one described previously. The main difference lies in the map data processing being performed outside of the models, more specifically, the neighboring cells utility function being called once every time a new individual is generated, with the resulting matrix stored in an ASCII file within each model folder, similarly to the individual map. The resulting structure of the model script is outlined below:

1. The first step is to load all the files as described in the previous model design. For compactness and compatibility, the input matrix is the cell combinations matrix instead of the individual map being evaluated.
2. The second step remains unchanged.
3. In the third step, the relevant helper model class is instantiated as previously described.
   a) For compactness and compatibility, the cell combinations are directly used to perform the respective calculations. There is no change with regards to the calculations in the economy model
   b) The final step involves updating the output file with the resulting score.

The three model designs were evaluated by performing two types of experiments, respectively, referred to as "small" and "large". This is explained by the difference in population size and the

**Table 4**
Relevant CoMOLA settings for each Experiment Type

| | Small | Large |
|---|---|---|
| **Number of Runs** | 10 | 10 |
| **Population Size** | 10 | 50 |
| **Max. Generations** | 2 | 10 |
| **Neighboring Cells** | 8 | 8 |
| **Mutation Rate** | 0.01 | 0.01 |
| **Crossover Rate** | 0.9 | 0.9 |

maximum generations set for each. The small experiment is attributed a population size of ten and two generations, whereas the large experiment has a population size of fifty and ten generations. This is much lower than the actual implementation of the source study [13], but was considered sufficient for the scope of this project. Other relevant algorithm configurations can be observed in Table 4. Finally, a set of ten runs was conducted for each test, providing initial insights while taking into account the time constraints of this study. All experimental code is available online [1]. The results are analyzed in the following section.

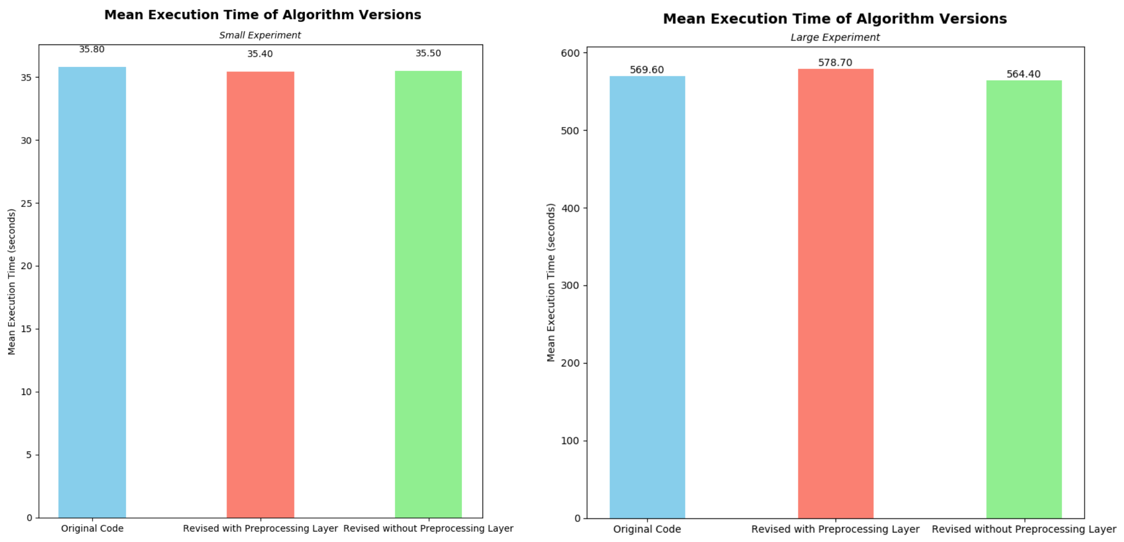


**Figure 3:** Mean execution time of each model design for the small experiment (left) and the large experiment (right)

**Table 5**
Results of the pair-wise statistical difference of mean execution times via the Turkey Test.

| group1 | group2 | meandiff | p-adj | lower | upper | reject |
|---|---|---|---|---|---|---|
| Original | Revised (P) | 9.1 | 0.001 | 5.0468 | 13.1532 | True |
| Original | Revised (NP) | -5.2 | 0.0099 | -9.2535 | -1.1468 | True |
| Revised (P) | Revised (NP) | -14.3 | 0.001 | -18.3532 | -10.2468 | True |

[1]https://github.com/Margauxvg/comola-thesis

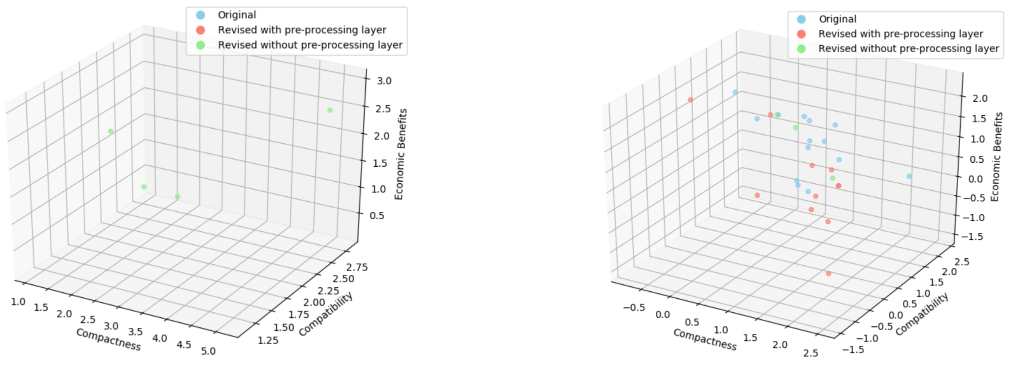


**Figure 4:** 3D scatter plot of the normalized fitness values of the set of Pareto-optimal solutions for the small experiment (left) and the large experiment (right).

## 3. Results

### 3.1. Time Performance

The time performance of the three model designs was compared using statistical analysis. For each experiment, the mean execution time of the ten runs was calculated and illustrated in Figure 3. Although a correlation between the increase in execution time and the increase in population size and generations can be observed, the plotting did not reveal a substantial variation in times. Therefore, an ANOVA test determined that there were significant differences between designs [29], but only within the large experiment. The absence of statistical significance in the small experiment is likely related to the generally very short execution times, compared to the large experiment. It is also plausible that larger population size and generation settings could lead to clearer variations of mean execution times.

A Turkey test was then used on the results of the large experiment in running time to identify pairs of means with statistical differences [29]. The outcomes are summarized in Table 5, revealing that all the models were statistically different from each other. Consequently, it was possible to discern the best-performing method, which appeared to be the revised version without a pre-processing layer.

It is important to note that the worst time performance is attributed to the revised version with a pre-processing layer. This is likely due to additional time required to format the neighboring cells matrix so that it is available for the models to use as input. As a matter of fact, the storage of new individual maps was mirrored to store the pre-processing layer output as follows:

1. The output matrix is converted to an ASCII format.
2. The converted matrix is then transferred to multiple files, one for each model. This is because each model has its own dedicated folder that contains all the necessary files for its execution.
3. The models then load the matrix and convert it back to a NumPy array before utilization.

As a consequence, the overhead introduced by these file operations may outweigh the advantages of performing map data processing only once for each generated individual.

### 3.2. Solution Quality Evaluation

#### 3.2.1. Objective Functions Optimization

At the end of every experiment run, the algorithm returns a set of Pareto-optimal solutions, in the form of land use maps and their corresponding scores of compactness maximization, compatibility maximization, and economic benefits maximization. In the section, the aim was to analyze whether a different model design as well as the integration of a preprocessing layer could surpass the original version of the CoMOLA tool in optimizing the considered land use management functions considered.

First, all the best solutions from each experiment were grouped into distinct sets. Due to notable differences in the model calculations between the original and revised implementations, a direct evaluation was not feasible. To address this and make a meaningful comparison, all scores were normalized using the z-score method, as shown in Equation 1.

$$Z = \frac{X - \mu}{\sigma} \tag{1}$$

- $Z$: Z-score
- $X$: Data point
- $\mu$: Mean of the dataset
- $\sigma$: Standard deviation of the dataset

With all values now standardized, the best solutions, regardless of the model type, were once again grouped into two distinct sets, one for each experiment. These sets no longer represent a Pareto-optimal set of solutions. To address this, a Pareto-optimality function was implemented to identify the new set of Pareto-optimal solutions for each experiment and to specify from which model design they originated. The results are illustrated in Figure 4.

The plots show clear differences between the small and large experiments. In the small experiment, the revised version without a pre-processing layer outperforms the other versions, exclusive in its contribution to the new set of Pareto-optimal solutions. In contrast, the large experiment includes all three model designs in the final set of best solutions, albeit in different proportions. The revised model without a pre-processing layer has the lowest representation, while the other two demonstrate similar levels, with a slight advantage for the original code. Therefore, it seems that the revised models improve the fitness performance in small experiment settings, but not necessarily in larger ones. Nevertheless, the presence of both revised models in the Pareto-optimal set of solutions for the large experiment, especially the substantial representation of the revised model with a preprocessing layer, indicates that the new model designs do not negatively affect the fitness performance.

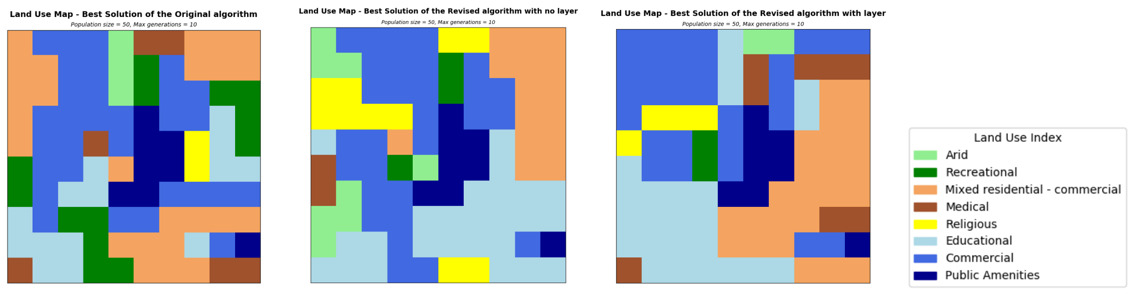

Land Use Map - Best Solution of the Original algorithm
Population size = 50, Max generations = 10
Land Use Map - Best Solution of the Revised algorithm with no layer
Population size = 50, Max generations = 10
Land Use Map - Best Solution of the Revised algorithm with layer
Population size = 50, Max generations = 10
Land Use Index
Arid
Recreational
Mixed residential - commercial
Medical
Religious
Educational
Commercial
Public Amenities

**Figure 5:** Selection of maps from the set of Pareto-optimal solution in the large experiment

Some considerations must be made with regard to these results. First, a different normalization method might have led to a different set of Pareto-optimal solutions. Second of all, the difference in representations of the two revised model designs seems to indicate that the version with a pre-processing layer outperforms the one without a pre-processing layer. However, the model calculations are the same. It is therefore possible that this difference is purely induced by the experimental settings and the degree of randomness involved in the generation of new individuals.

### 3.2.2. Solution Maps

In the previous section, the set of Pareto-optimal solutions was established for each experiment. Given that the three types of model design were present in the large experiment's set, the large experiment was chosen for a visual comparison between the status quo map (Figure 2) and the best solutions. To simplify this comparison, one solution for each type of model was extracted and converted back into a colored land use map, as illustrated in Figure 5.

On all solution maps, changes can be observed in line with the experiment constraints as well as with the different objective functions. For instance, Public Amenities have remained unchanged. More parcels are allocated to land uses contributing to economic benefits, respectively Mixed residential-commercial and Commercial. In addition, there is a noticeable increase in the clustering of land uses in patches, which aligns with the maximization of compactness and compatibility, since neighboring cells are most compatible when sharing the same land use.

The proposal for a set of Pareto-optimal set of solutions does not allow for the identification of one unique best solution for this optimization problem, as tempting as it may be. Although there are existing methods to rank Pareto-optimal solutions [30], ultimately the decision remains in the hands of policy makers and domain experts. In fact, CoMOLA is not implemented to exactly mirror reality, but often uses a simplified version of land use maps and land use management functions. In real-life scenarios, additional environmental, social and economic aspects must be taken into account when selecting changes in land use allocation that would best fit the area of interest [5, 17].

## 4. Discussion

### 4.1. Limitations

Although the results are promising, the experiments performed and the objective functions tested stemmed from one specific research paper utilizing CoMOLA. The three land use management functions implemented were also rather straightforward, with two relying solely on map-extracted data. These particularities might limit the direct applicability of the proposed model designs to more varied and intricate optimization problems. In order to enhance confidence in the robustness and versatility of abstract model formatting within CoMOLA, a broader spectrum of research should be experimented with, with an emphasis on adapting more complex objective functions with the new design.

Moreover, the performance of the optimization algorithm, both in terms of execution time and solution quality, can be influenced by how the user configures the tool. For instance, the outcomes of the fitness functions depend on the input map, the surface and transition constraints, which serve as the basis for generating improved land use maps. For consistency purposes, these settings were adopted from the original research paper and left unchanged in all experiments conducted. However, altering the input map or constraints might lead the objective functions to evaluate the individuals differently, ultimately proposing a different set of Pareto-optimal solutions and potentially favoring a different model design than the one suggested by the results analysis. Also, there seems to be a consensus in the reviewed literature on the relatively poor time performance of the CoMOLA tool, which worsens as generations increase. For the scope of this study, while some configurations remained consistent with the source study, others were reduced, such as population size and generations. In addition, only a limited number of runs were performed for each experiment. Therefore, a more comprehensive exploration of the tool's settings and a larger number of runs are essential to improve the generalizability of the findings across diverse optimization scenarios.

Furthermore, due to the recent introduction of the CoMOLA tool in the research community, a limited number of studies were reviewed. This lack of a diverse literature base restricted the range of perspectives considered in the development of the abstracted land use management model and ready-to-use implementations. A greater use of the tool in various land use management optimization contexts would offer the opportunity to validate, adapt, or extend the proposed model design.

Finally, it is important to mention that the ontological representation introduced in this study has not received approval from domain experts. Additionally, its potential benefits remain theoretical, as it was neither implemented nor integrated within the CoMOLA tool, nor was it tested. Despite the study's aim to present a framework for enhancing knowledge representation and language unification in CoMOLA, this may raise questions about its suitability and relevance in practical applications.

### 4.2. Conclusions

The primary focus of this study was to explore the impact of integrating knowledge representation techniques into the model design of optimization algorithms, specifically within the CoMOLA land use management optimization tool. The results highlight that abstract modeling can significantly reduce the programming challenges for users, without damaging the extensibility of the objective functions. This user-centric approach aligns with the practical needs of domain experts, empowering them to apply their specialized knowledge, in the form of inputs, constraints, and external data sources that can be utilized within the models. The separation of common functionalities for all CoMOLA implementations reviewed, such as map data processing, into shared files and utility functions appears to be a powerful strategy for code readability, maintainability, and performance. Between the two model designs proposed, one with a pre-processing layer and one without, some differences in performance can be observed. In fact, although the preprocessing layer offers a more streamlined implementation and shows superiority in the fitness evaluation, it also negatively affects the execution time of the optimization process. The integration of a preprocessing layer would therefore be more relevant in optimization problems, where a multitude of objective functions treat similar map-extracted data. By providing access to both model designs, users could select a preferred modeling method, further tailoring CoMOLA to their research needs. Furthermore, the successful testing of the revised models showcases the capabilities of the new formatting in real-world applications in land use management and related domains. If CoMOLA was originally designed for land use management optimization, an abstracted version of the tool could be beneficial to other research contexts. For instance, although land use indexes currently identify the cells of a map, these could very well be interpreted differently, for example, as plant types when optimizing the composition and configuration of a garden. Finally, the ontological framework could become the key to offering a standardized language that could improve interoperability and foster a more cohesive understanding of land use management.

As mentioned earlier, future research should encompass further validation and refinement of the proposed designs in more diverse and more complex land use management scenarios. Analyzing more literature implementing similar objective functions should also help with the development of more utility functions as well as more general and simplified models. In order to maintain and potentially increase the extensibility of the tool, the model layout should be improved to allow the integration of multiple external data sources as well as different types of external sources. If abstracted formatting was confirmed as an asset for the integration of models within CoMOLA, other optimization algorithms requiring user-provided models could be revised with similar structures. Moreover, other components of the CoMOLA tool could be revised with the goal of enhancing its performance. For example, addressing the current time performance issue might involve refactoring the code to mitigate computationally intensive data storage. Following the validation by domain experts of a suitable ontological framework, evaluating its integration and effectiveness as a domain knowledge preprocessing method should be considered. Furthermore, the creation of a more user-friendly interface could further reduce the programming load on users. For example, such an interface could only require the necessary inputs from domain experts, including the selection of objective functions to consider. The tool could then perform the optimization in the background and provide the user with a report containing the algorithm's output data, already processed in the form of graphs. Finally, since CoMOLA proves to be a powerful tool, its applicability in domains other than land use management optimization should be investigated.

In essence, this study underscores the benefits of collaboration between domain experts and computational scientists to effectively support decision-makers. The path forward involves continuous improvement of optimization algorithms such as CoMOLA to ensure that they evolve to meet the ever-growing demands of complex problem solving in land use management and other applications.

## 5. Acknowledgements

I express my gratitude to Stefan Schlobach as main supervisor, for his invaluable counsel and support throughout the project, and to Benno Kruit as second reader, for his thoughtful feedback and detailed evaluation of the thesis. Thanks to Žiga Malek, former environmental sciences professor at the Vrije Universiteit, for his initial guidance. The authors thank researchers Sarah Mikaeil and Nina Schwarz for sharing their CoMOLA implementations, improving the comparative analysis of objective functions, and testing the new proposed model design. Special thanks to Michael Strauch, developer of the CoMOLA tool, for his time and insights, significantly contributing to this project's depth and quality.

This project has benefited enormously from the collective expertise and support of these individuals, and their contributions are greatly appreciated.

## References

[1] H. Geerlings, D. Stead, The integration of land use planning, transport and environment in european policy and research, Transport Policy 10 (2003) 187–196. URL: https://www.sciencedirect.com/science/article/pii/S0967070X03000209. doi:https://doi.org/10.1016/S0967-070X(03)00020-9, urban Transport Policy Instruments.

[2] S. Garg, Impact of Overpopulation on Land Use Pattern, 2017. doi:10.4018/978-1-5225-1683-5.ch008.

[3] B. Entwisle, P. C. Stern (Eds.), Population, Land Use, and Environment: Research Directions, National Academies Press (US), Washington, DC, 2005. Panel Report.

[4] Q. Tong, F. Qiu, Population growth and land development: Investigating the bi-directional interactions, Ecological Economics 169 (2020) 106505. URL: https://www.sciencedirect.com/science/article/pii/S0921800919302708. doi:https://doi.org/10.1016/j.ecolecon.2019.106505.

[5] A. Kaim, M. Strauch, M. Volk, Using stakeholder preferences to identify optimal land use configurations, Frontiers in Water 2 (2020). URL: https://www.frontiersin.org/articles/10.3389/frwa.2020.579087. doi:10.3389/frwa.2020.579087.

[6] L. M. Scott, M. V. Janikas, Spatial statistics in arcgis, in: Handbook of applied spatial analysis: Software tools, methods and applications, Springer, 2009, pp. 27–41.

[7] J. van Vliet, P. H. Verburg, A Short Presentation of CLUMondo, Springer International Publishing, Cham, 2018, pp. 485–492. URL: https://doi.org/10.1007/978-3-319-60801-3_34. doi:10.1007/978-3-319-60801-3_34.

[8] P. Peltonen-Sainio, L. Jauhiainen, H. Laurila, J. Sorvali, E. Honkavaara, S. Wittke, M. Karjalainen, E. Puttonen, Land use optimization tool for sustainable intensification of high-latitude agricultural systems, Land Use Policy 88 (2019) 104104. URL: https://www.sciencedirect.com/science/article/pii/S0264837718319781. doi:https://doi.org/10.1016/j.landusepol.2019.104104.

[9] A. Shaamala, T. Yigitcanlar, A. Nili, D. Nyandega, Algorithmic green infrastructure optimisation: Review of artificial intelligence driven approaches for tackling climate change, Sustainable Cities and Society 101 (2024) 105182. URL: https://www.sciencedirect.com/science/article/pii/S221067072400012X. doi:https://doi.org/10.1016/j.scs.2024.105182.

[10] M. Strauch, C. Pätzold, Comola: Comprehensive multi-objective land use allocation, 2019. URL: https://github.com/michstrauch/CoMOLA/blob/master/README.md, accessed: .

[11] M. Strauch, A. F. Cord, C. Pätzold, S. Lautenbach, A. Kaim, C. Schweitzer, R. Seppelt, M. Volk, Constraints in multi-objective optimization of land use allocation – repair or penalize?, Environmental Modelling & Software 118 (2019) 241–251. URL: https://www.sciencedirect.com/science/article/pii/S1364815218311204. doi:https://doi.org/10.1016/j.envsoft.2019.05.003.

[12] Y. Yusoff, M. S. Ngadiman, A. M. Zain, Overview of nsga-ii for optimizing machining process parameters, Procedia Engineering 15 (2011) 3978–3983. URL: https://www.sciencedirect.com/science/article/pii/S1877705811022466. doi:https://doi.org/10.1016/j.proeng.2011.08.745, cEIS 2011.

[13] S. M. Abdel-Ghany, H. M. Ayad, I. A. Elcherif, Sustainable multi-objective optimisation in land-use planning based on non-dominated sorting genetic algorithm (nsga-ii): a case study in alexandria, egypt, Mobility, Knowledge and Innovation Hubs in Urban and Regional Development. Proceedings of REAL CORP 2022, 27th International Conference on Urban Development, Regional Planning

and Information Society (2022) 733–744.

[14] N. Schwarz, F. Hoffmann, S. Knapp, M. Strauch, Synergies or trade-offs? optimizing a virtual urban region to foster plant species richness, climate regulation, and compactness under varying landscape composition, Frontiers in Environmental Science 8 (2020). URL: https://www.frontiersin.org/articles/10.3389/fenvs.2020.00016. doi:10.3389/fenvs.2020.00016.

[15] V. D. Picasso, The "biodiversity–ecosystem function debate": An interdisciplinary dialogue between ecology, agricultural science, and agroecology, Agroecology and Sustainable Food Systems 42 (2018) 264–273. URL: https://doi.org/10.1080/21683565.2017.1359806. doi:10.1080/21683565.2017.1359806. arXiv:https://doi.org/10.1080/21683565.2017.1359806.

[16] W. Verhagen, E. H. van der Zanden, M. Strauch, A. J. van Teeffelen, P. H. Verburg, Optimizing the allocation of agri-environment measures to navigate the trade-offs between ecosystem services, biodiversity and agricultural production, Environmental Science & Policy 84 (2018) 186–196. URL: https://www.sciencedirect.com/science/article/pii/S1462901117310018. doi:https://doi.org/10.1016/j.envsci.2018.03.013.

[17] B. Bartkowski, M. Beckmann, M. Drechsler, A. Kaim, V. Liebelt, B. Müller, F. Witing, M. Strauch, Aligning agent-based modeling with multi-objective land-use allocation: Identification of policy gaps and feasible pathways to biophysically optimal landscapes, Frontiers in Environmental Science 8 (2020). URL: https://www.frontiersin.org/articles/10.3389/fenvs.2020.00103. doi:10.3389/fenvs.2020.00103.

[18] P. P. Bonissone, R. Subbu, N. Eklund, T. R. Kiehl, Evolutionary algorithms+ domain knowledge= real-world evolutionary computation, IEEE Transactions on Evolutionary Computation 10 (2006) 256–280.

[19] A. Gellert, A. Florea, U. Fiore, P. Zanetti, L. Vintan, Performance and energy optimisation in cpus through fuzzy knowledge representation, Information Sciences 476 (2019) 375–391. URL: https://www.sciencedirect.com/science/article/pii/S0020025518302068. doi:https://doi.org/10.1016/j.ins.2018.03.029.

[20] P. Benjamin, M. Patki, R. Mayer, Using ontologies for simulation modeling, in: Proceedings of the 2006 Winter Simulation Conference, 2006, pp. 1151–1159. doi:10.1109/WSC.2006.323206.

[21] M. Katsumi, Land use ontology, ???? URL: http://ontology.eil.utoronto.ca/icity/LandUse/1.2, revision: 1.2.

[22] A. Azad, X. Wang, Land use change ontology and traffic prediction through recurrent neural networks: A case study in calgary, canada, ISPRS International Journal of Geo-Information 10 (2021). URL: https://www.mdpi.com/2220-9964/10/6/358. doi:10.3390/ijgi10060358.

[23] H. Silvennoinen, A. Chadzynski, F. Farazi, A. Grišiūtė, Z. Shi, A. von Richthofen, S. Cairns, M. Kraft, M. Raubal, P. Herthogs, A semantic web approach to land use regulations in urban planning: The ontozoning ontology of zones, land uses and programmes for singapore, Journal of Urban Management 12 (2023) 151–167. URL: https://www.sciencedirect.com/science/article/pii/S2226585623000067. doi:https://doi.org/10.1016/j.jum.2023.02.002.

[24] N. Chichkova, A. Begler, V. Vlasov, Modeling city land use with an ontology, 2020. doi:10.1145/3428502.3428638.

[25] A. Planning Association, Land-based classification standards - lbcs tables, 2001. URL: https://planning-org-uploaded-media.s3.amazonaws.com/document/LBCS.pdf.

[26] A. R. Harrison, National land use database: Land use and land cover classification version 4.4, 2006. Report prepared for the Office of the Deputy Prime Minister.

[27] Eurostat, the statistical office of the European Union, Land cover and land use, landscape (LUCAS) (lan), Eurostat, the statistical office of the European Union, European Commission - Eurostat - L-2920 LUXEMBOURG, 2017. URL: https://ec.europa.eu/eurostat/cache/metadata/en/lan_esms.htm, reference Metadata in Euro SDMX Metadata Structure (ESMS).

[28] K. Gülen, What is abstraction in computer science?, 2023. URL: https://dataconomy.com/2023/03/31/what-is-abstraction-in-computer-science/, accessed: .

[29] H. Abdi, L. J. Williams, Newman-keuls test and tukey test, Encyclopedia of research design 2 (2010) 897–902.

[30] R. Rao, R. Lakshmi, Ranking of pareto-optimal solutions and selecting the best solution in multi- and many-objective optimization problems using r-method, Soft Computing Letters 3 (2021) 100015. URL: https://www.sciencedirect.com/science/article/pii/S2666222121000058. doi:https://doi.org/10.1016/j.socl.2021.100015.

## A. Large Figures

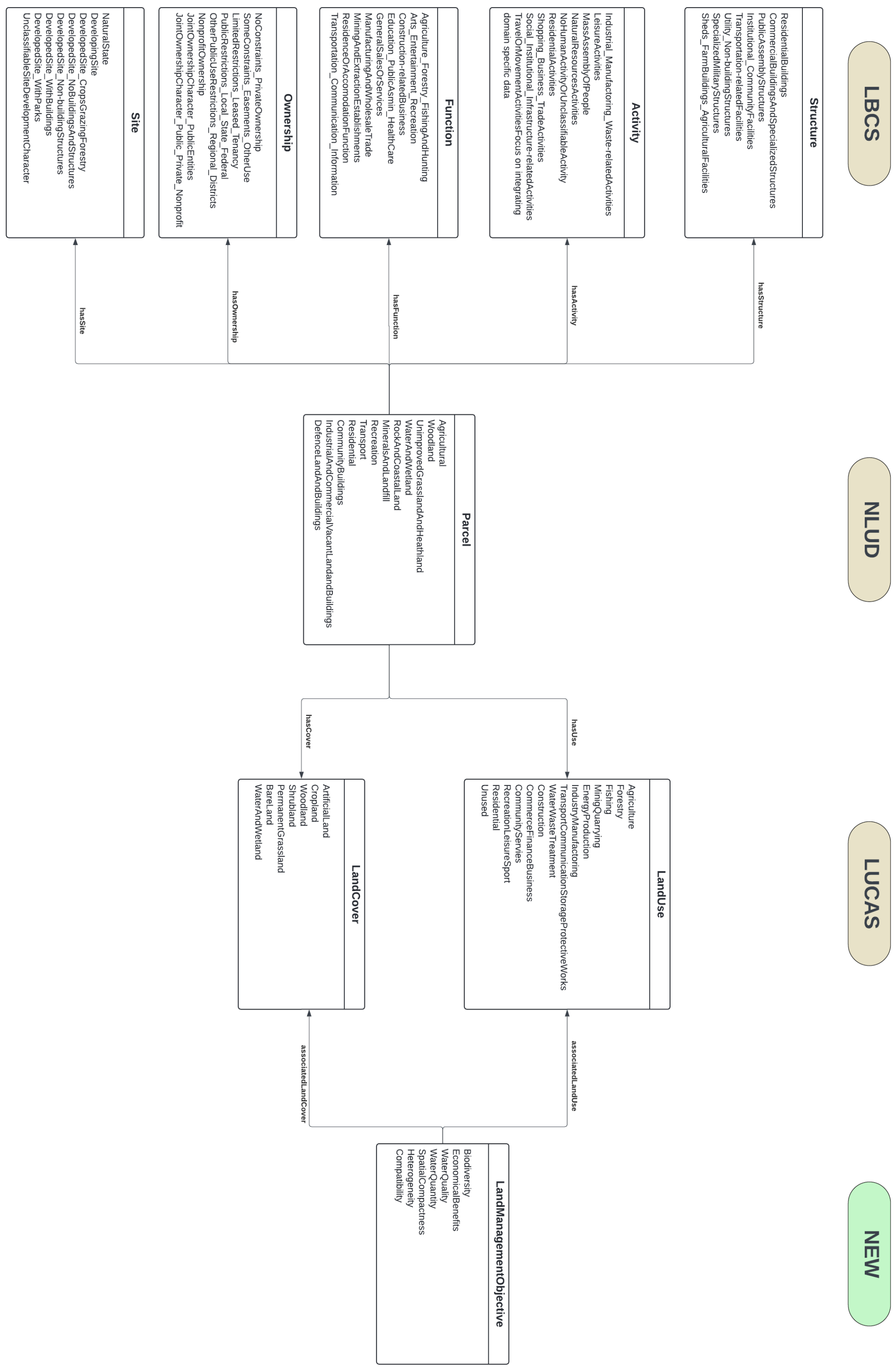
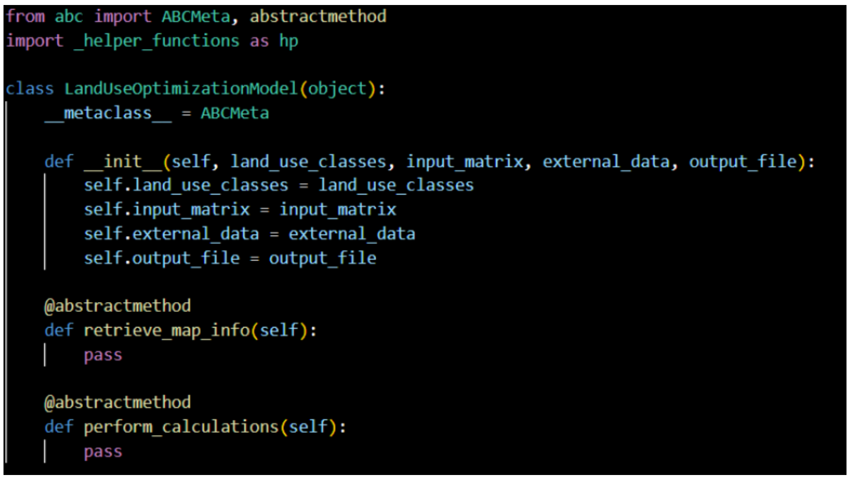
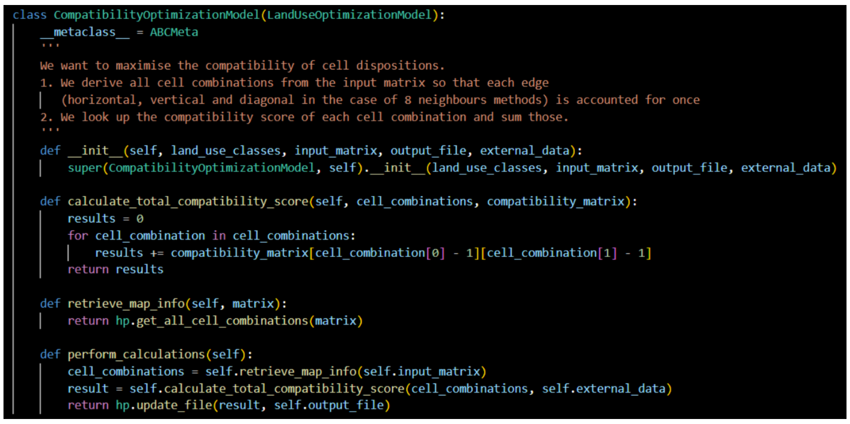

LBCS
Structure
ResidentialBuildings
CommercialBuildingsAndSpecializedStructures
PublicAssemblyStructures
Institutional_CommunityFacilities
Transportation-relatedFacilities
Utility_Non-buildingStructures
SpecializedMilitaryStructures
Sheds_FarmBuildings_AgriculturalFacilities
Activity
Industrial_Manufactoring_Waste-relatedActivities
LeisureActivities
MassAssemblyOfPeople
NaturalResourcesActivities
NoHumanActivityOrUnclassifiableActivity
ResidentialActivities
Shopping_Business_TradeActivities
Social_Institutional_Infrastructure-relatedActivities
TravelOrMovementActivitiesFocus on integrating domain specific data
Function
Agriculture_Forestry_FishingAndHunting
Arts_Entertainment_Recreation
Construction-relatedBusiness
Education_PublicAsmin_HealthCare
GeneralSalesOrServices
ManufacturingAndWholesaleTrade
MiningAndExtractionEstablishments
ResidenceOrAccomodationFunction
Transportation_Communication_Information
Ownership
NoConstraints_PrivateOwnership
SomeConstraints_Easements_OtherUse
LimitedRestrictions_Leased_Tenancy
PublicRestrictions_Local_State_Federal
OtherPublicUseRestrictions_Regional_Districts
NonprofitOwnership
JointOwnershipCharacter_PublicEntities
JointOwnershipCharacter_Public_Private_Nonprofit
Site
NaturalState
DevelopingSite
DevelopedSite_CropsGrazingForestry
DevelopedSite_NoBuildingsAndStructures
DevelopedSite_Non-buildingStructures
DevelopedSite_WithBuildings
DevelopedSite_WithParks
UnclassifiableSiteDevelopmentCharacter
hasSite
hasOwnership
hasFunction
hasActivity
hasStructure
NLUD
Parcel
Agricultural
Woodland
UnimprovedGrasslandAndHeathland
WaterAndWetland
RockAndCoastalLand
MineralsAndLandfill
Recreation
Transport
Residential
CommunityBuildings
IndustrialAndCommercialVacantLandandBuildings
DefenceLandAndBuildings
hasCover
hasUse
LUCAS
LandUse
Agriculture
Forestry
Fishing
MinigQuarrying
EnergyProduction
IndustryManufactoring
TransportCommunicationStorageProtectiveWorks
WaterWasteTreatment
Construction
CommerceFinanceBusiness
CommunityServies
RecreationLeisureSport
Residential
Unused
LandCover
ArtificialLand
Cropland
Woodland
Shrubland
PermanentGrassland
BareLand
WaterAndWetland
associatedLandUse
associatedLandCover
NEW
LandManagementObjective
Biodiversity
EconomicalBenefits
WaterQuality
WaterQuantity
SpatialCompactness
Heterogeneity
Compatibility

**Figure 6:** Extended Ontological Classification for Land Use Management



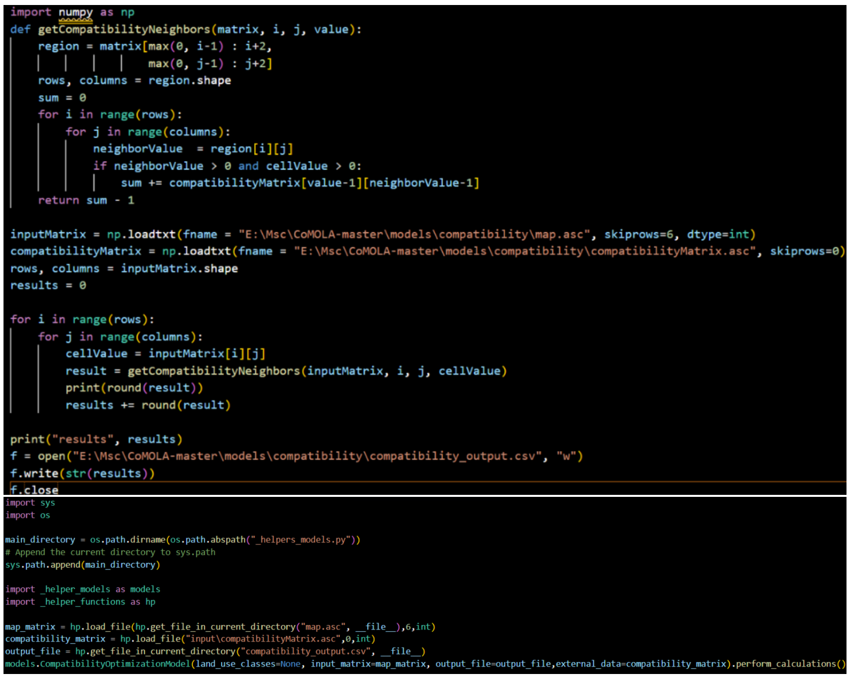
```python
from abc import ABCMeta, abstractmethod
import _helper_functions as hp

class LandUseOptimizationModel(object):
    __metaclass__ = ABCMeta

    def __init__(self, land_use_classes, input_matrix, external_data, output_file):
        self.land_use_classes = land_use_classes
        self.input_matrix = input_matrix
        self.external_data = external_data
        self.output_file = output_file

    @abstractmethod
    def retrieve_map_info(self):
        pass

    @abstractmethod
    def perform_calculations(self):
        pass
```

**Figure 7:** Code snippet of the Land Use Optimization Model class



```python
class CompatibilityOptimizationModel(LandUseOptimizationModel):
    __metaclass__ = ABCMeta
    '''
    We want to maximise the compatibility of cell dispositions.
    1. We derive all cell combinations from the input matrix so that each edge
       (horizontal, vertical and diagonal in the case of 8 neighbours methods) is accounted for once
    2. We look up the compatibility score of each cell combination and sum those.
    '''
    def __init__(self, land_use_classes, input_matrix, output_file, external_data):
        super(CompatibilityOptimizationModel, self).__init__(land_use_classes, input_matrix, output_file, external_data)

    def calculate_total_compatibility_score(self, cell_combinations, compatibility_matrix):
        results = 0
        for cell_combination in cell_combinations:
            results += compatibility_matrix[cell_combination[0] - 1][cell_combination[1] - 1]
        return results

    def retrieve_map_info(self, matrix):
        return hp.get_all_cell_combinations(matrix)

    def perform_calculations(self):
        cell_combinations = self.retrieve_map_info(self.input_matrix)
        result = self.calculate_total_compatibility_score(cell_combinations, self.external_data)
        return hp.update_file(result, self.output_file)
```

**Figure 8:** Code snippet of the Compatibility Optimization Model class



```python
import numpy as np
def getCompatibilityNeighbors(matrix, i, j, value):
    region = matrix[max(0, i-1) : i+2,
                    max(0, j-1) : j+2]
    rows, columns = region.shape
    sum = 0
    for i in range(rows):
        for j in range(columns):
            neighborValue  = region[i][j]
            if neighborValue > 0 and cellValue > 0:
                sum += compatibilityMatrix[value-1][neighborValue-1]
    return sum - 1

inputMatrix = np.loadtxt(fname = "E:\Msc\CoMOLA-master\models\compatibility\map.asc", skiprows=6, dtype=int)
compatibilityMatrix = np.loadtxt(fname = "E:\Msc\CoMOLA-master\models\compatibility\compatibilityMatrix.asc", skiprows=0)
rows, columns = inputMatrix.shape
results = 0

for i in range(rows):
    for j in range(columns):
        cellValue = inputMatrix[i][j]
        result = getCompatibilityNeighbors(inputMatrix, i, j, cellValue)
        print(round(result))
        results += round(result)

print("results", results)
f = open("E:\Msc\CoMOLA-master\models\compatibility\compatibility_output.csv", "w")
f.write(str(results))
f.close
```

```python
import sys
import os

main_directory = os.path.dirname(os.path.abspath("_helpers_models.py"))
# Append the current directory to sys.path
sys.path.append(main_directory)

import _helper_models as models
import _helper_functions as hp

map_matrix = hp.load_file(hp.get_file_in_current_directory("map.asc", __file__),6,int)
compatibility_matrix = hp.load_file("input\compatibilityMatrix.asc",0,int)
output_file = hp.get_file_in_current_directory("compatibility_output.csv", __file__)
models.CompatibilityOptimizationModel(land_use_classes=None, input_matrix=map_matrix, output_file=output_file,external_data=compatibility_matrix).perform_calculations()
```

**Figure 9:** Code snippets of model files for the compatibility objective function. The original version is at the top [13]. The revised version is at the bottom.