Dynamic Programming for Aligning Sketch Maps

Dissertation submitted in partial fulfillment of the requirements

for the Degree of Master of Science in Geospatial Technologies

February 24, 2020

Violeta Ana Luz Sosa León

vsosaleo@uni-muenster.com

https://github.com/violetasdev

Supervised by:

Prof. Dr. Angela Schwering

Institute for Geoinformatics

University of Münster

Co-supervised by:

Dr. Malumbo Chipofya

Institute for Geoinformatics

University of Münster

and

Prof. Dr. Marco Painho

Nova Information Management School

Universidade Nova de Lisboa

ifgi
Institute for Geoinformatics
University of Münster

NOVA IMS Information Management School

UNIVERSITAT JAUME I

Declaration of Academic Integrity

I hereby confirm that this thesis on *Dynamic Programming for Aligning Sketch Maps* is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited.

February 24, 2020

I agree to have my thesis checked in order to rule out potential similarities with other works and to have my thesis stored in a database for this purpose.

February 24, 2020

Acknowledgments

I am grateful to the joint program coordinated by Dr. Christoph Brox at the University of Muenster, Prof. Dr. Joaquín Huerta at Jaume I University, and my also supervisor, Prof. Dr. Marco Painho at the University Nova de Lisboa for this opportunity to have one of the most significant experiences in my academic, social, and professional aspects. I hope you can continue helping more students in the future with the program from which I have been part.

A special thanks to my supervisors, Prof. Dr. Angela Schwering and Dr. Malumbo Chipofya, from whom I have learned in a range of different topics from theoretical to technical, in how to engage research and to manage myself to improve continually. Your kind support and knowledge have made possible this thesis to flourish and be an essential step in my future academic and professional life. I am inspired and motivated to be more involved in the Spatial Intelligence research field and continuing exploring the interdisciplinarity approaches between spatial cognition and computer science.

Finally, I would like to acknowledge my family, professors, and friends in Mexico, Colombia, and Japan, and my new friends all over the world. Thanks for your overwhelming care and support in this adventure, pushing me out of my comfort zone to learn from every experience and giving me love, courage, and advice when I most needed it.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Algorithms

List of Acronyms

DPSM	Dynamic Programming Sketch Maps Implementation
FP	False Positive
FN	False Negative
GIS	Geographic Information System
LA	Link Analysis
LCM	Local Compatibility Matrix
LCM(H1)	First heuristic score from the LCM
LCM(H2)	Second heuristic score from the LCM
QMC	Qualitative Constraint Map
QCN	Qualitative Constraint Network
SST	Spectral Solution Technique
TP	True Positive
TN	True Negative

Abstract

Sketch maps play an important role in communicating spatial knowledge, particularly in applications interested in identifying correspondences to metric maps for land tenure in rural communities. The interpretation of a sketch map is linked to the users' spatial reasoning and the number of features included. Additionally, in order to make use of the information provided by sketch maps, the integration with information systems is needed but is convoluted. The process of identifying which element in the base map is being represented in the sketch map involves the use of correct descriptors and structures to manage them. In the past years, different methods to give a solution to the sketch matching problem employs iterative methods using static scores to create a subset of correspondences. In this thesis, we propose an implementation for the automatic aligning of the sketch to metric maps, based on dynamic programming techniques from reinforcement learning. Our solution is distinctive from other approaches as it searches for pair equivalences by exploring the environment of the search space and learning from positive rewards derived from a custom scoring system. Scores are used to evaluate the likeliness of a candidate pair to belong to the final solution, and the results are back up in a state-value function to recover the best subset states and recovering the highest scored combinations. Reinforcement learning algorithms are dynamic and robust solutions for finding the best solution in an ample search space. The proposed workflow improves the outcoming spatial configuration for the aligned features compared to previous approaches, specifically the Tabu Search.

Keywords: sketch map, metric map, dynamic programming, tabu search, learning algorithm, link analysis, alignment.

CHAPTER 1 INTRODUCTION

1.1 RELATED WORK AND MOTIVATION

As humans, we communicate our perception of the elements surrounding us by using different tools: book descriptions, paintings, and more elaborated representations like maps in different types, including sketch maps. Sketch maps contain a set of items displaying the author's conception of the space, providing spatial information useful for studying and understanding the environment in which she lives (Malumbo Chipofya, Wang, & Schwering, 2011). To be able to unveil the meaning behind a sketch map without the author's feedback, it is necessary to compare every structure to a more structured representation of geographical elements, such as metric maps (Klaus Broelemann, Jiang, & Schwering, 2016). By having them side-by-side, it is possible to identify the abstraction created in the sketched map and relate it to a specific item in the metric map. As the elements increase in the input map, the association's complexity with the metric map also grows, and therefore the relationships included, requiring automatizing the *aligning* process.

The difficulties in this task include the definition of appropriate representations of the problem space in order to structure the search for correspondences (Wallgrün, Wolter, & Richter, 2010). Graphs are robust information structures with gained popularity to represent formal structures for displaying relations of different types such as spatial, geometrical, or conceptual (Bunke, 2000). They are often used to examine the relationships correspondence and consistency of the data structure implemented with an exhaustive analysis of their distribution defined as a case of graph matching problems, with different approaches according to the category in which the graph representation belongs (Foggia, Percannella, & Vento, 2014). Diverse techniques to solve the matching problem include the measure of distances, composite graph similarities, string-based methods, and statistical graph matching (Emmert-Streib, Dehmer, & Shi, 2016). One of the current implementations for the sketch to the metric alignment problem, translated as a graph matching problem with the implementation of Qualitative Constraint

Networks, analyzes specialized local structures to evaluate candidate pairs while searching for correspondences (Malumbo Chipofya, Schwering, & Binor, 2013). The correspondence problem using LCM has arisen solutions with exponential time complexity, which may not be a feasible solution for significant scale problems (Malumbo Chipofya, 2018). In the recent years, the artificial intelligence field has developed different techniques for giving solutions to large scale tasks involved with graph nature problems in computer vision, integrating algorithms that rely on patterns and deductions from the accessible information (Foggia et al., 2014). By exploiting the capabilities of Local Compatibility Matrices, newer algorithms for significant scale problems, and other similarities measures studied for matching tasks in other fields, *how to improve the pair selection process by taking advantage of past exploration in local compatibility matrices?*

1.2 RESEARCH QUESTIONS AND OBJECTIVES

This research aims to implement and compare two searching algorithms to identify the next optimal pair selection during the matching process between a sketch and a metric map. The following research questions are defined:

- *How can the pair selection algorithm be modified to increase the number of correctly matching objects for alignment between sketch and metric maps?*
- *How can the exploration in the pair selection algorithm be used to recover critical information for the matching process between sketch and metric maps?*
- *Does the new pair selection algorithm improve the alignment solution?*

In order to answer the previous research questions, the following objectives are defined:

- *Calculate a new selection score system for the matching process*
- *Retain feedback for future decision processes during the exploration in the search algorithm*
- *Evaluate the matching results comparing the search algorithms implemented to measure changes in performance*

The open challenges identified from the sketch to metric map alignment process are addressed with the stated research questions as displayed in Figure 1:

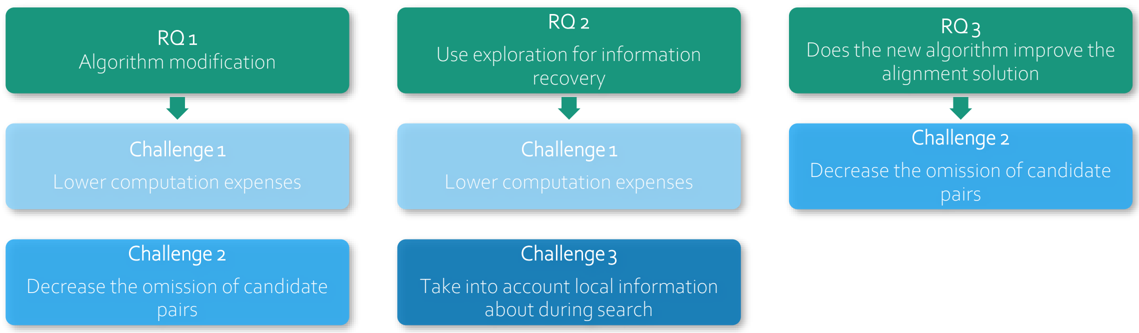

RQ 1
Algorithm modification
Challenge 1
Lower computation expenses
Challenge 2
Decrease the omission of candidate pairs
RQ 2
Use exploration for information recovery
Challenge 1
Lower computation expenses
Challenge 3
Take into account local information about during search
RQ 3
Does the new algorithm improve the alignment solution
Challenge 2
Decrease the omission of candidate pairs

Figure 1 Thesis challenges outline

1.3 GENERAL METHODOLOGY

The sketch to metric map alignment process workflow is divided into 5 modules: first of all, for the *Input Processing*, the sketch and metric map are processed in the SmartSkeMa framework to translate the features from geometries to a set of vectors by a computer vision segmentation process and then identify the spatial relationships from the vectorized features are organized in a graph like data structure, implementing Qualitative Constraint Networks. Secondly, the *Qualitative Analysis* module analyzes the output from the framework and assesses the compatibility of each feature in terms of feature type and similarity. The next step in the workflow is the *Score System*, which provides the measurements of Link Analysis ranking score, Spectral Solution clustering solution, and the two Local Compatibility Matrices Heuristic Scores H1 and H2 to evaluate the likeliness of a candidate pair to belong to the alignment solution. Finally, in the *Searching Algorithms* component, we implement two different algorithms to find correspondences between candidate pairs: a metaheuristic approach named the Tabu Search and a reinforcement learning algorithm, SARSA. We compare the provided solutions in terms of their performance, precision, and recall. Figure 2 summarizes the processes outlined and their outputs.

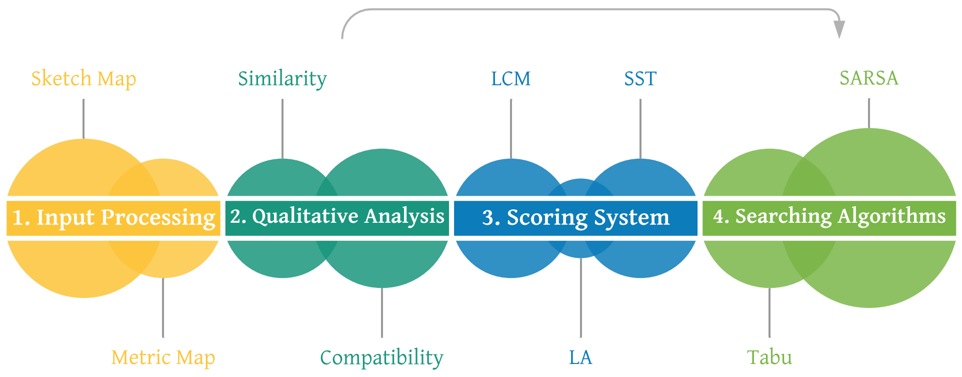


Figure 2 Methodology overview

1.4 THESIS OUTLINE

The following sections are organized as follows:

- **Chapter 2** describes the theoretical background introducing the concepts employed in this thesis including the use of sketch maps, the importance and past work for giving solution to the graph matching problem, the spatial representation calculus and data structures implemented to make use of them, and finally, a review to different strategies for searching algorithms.
- In **Chapter 3,** the proposed methodology is outlined in detail, describing the sketch and metric maps used as an input and the assessment of the compatibility between features followed by the definition of the scoring system for the pair selection, and finally, the design and pseudo code for the Tabu and SARSA algorithms.
- **Chapter 4** displays the results of the workflow described in Chapter 3 to automatically align sketch maps followed by the evaluation process to measure the performance and quality of the solution in terms of precision and recall for each search algorithm, discussion of the results, as well as the findings and the encountered limitations.
- Finally, **Chapter 5** includes the conclusions and future work.

CHAPTER 2 BACKGROUND

This section presents an outline of the concepts supporting this thesis from the literature. Initially, we present Sketch Maps and a brief background to the graph matching problem, introducing qualitative spatial representations and the different qualitative calculi involved. Secondly, we outline the theory of Link Analysis to identify the importance of vicinities in local exploration, and finally, we portray strategies to solve searching problems from the perspective of artificial intelligence.

2.1 SKETCH MAPS

Sketch maps are representations of the space surrounding an individual decomposed into different spatial elements such as roads, buildings, and other physical features describing the relationships between the scene elements (Schwering & Wang, 2010). Moreover, every individual due to different experiences give an interpretation of the objects and their relationships being a topic of interest in research for map sketching in schools, governmental projects, and academia.

The decoding of the information from a sketch (input) map to a metric (output) map is an approach by projects such as the SmartSkeMa framework delivering the scene spatial segmentation, qualitative representations, and input/output alignment process (Schwering et al., 2014). Along with the implementation, several analyses and techniques have been implemented to give solutions to the alignment process resulting in theoretical implications and findings such as Qualitative Constraint Matrices. On the other hand, the its4land project is one of the real-world applications of this kind of framework. By using sketch maps, communities in Kenya are able to participate in land delimitation and appropriation, helping to the management of natural and human-build resources.

To accurate relate subjective maps and metric maps, techniques for assessing qualitative map alignment has been applied to find matches among the input representation such as a sketched entity, and one or several entities in a metric map using Local Compatibility Matrices (LCM) (M. C. Chipofya, Schultz, & Schwering, 2016). The version of the implemented Tabu algorithm aims to face challenges such as the omission of promissory matching candidates and long execution times on large datasets. However, the dynamic metaheuristics generated in Chipofya's algorithm gave better performance and accuracy versus standard compatibility matrices; leaving open the research for the refinement during the iterative match-candidates selection process since it rapidly leaves the matching process without candidates due to the removal of not compatible local pairs at a particular stage. As the search space grows, it may be helpful to identify how to associate potential additions during the exploration in previous regions of the search space (Malumbo Chipofya et al., 2013)

2.2 THE GRAPH MATCHING PROBLEM

The most significant benefit of graphs is that they can represent structured data and have been used to undertake problems in data mining, document analysis, and graphical pattern recognition, and bioinformatics (Cook & Holder, 2006). A graph $g = (V, E, \alpha, \beta)$ is composed by:

- V, a set of finite nodes
- $E \subseteq V \times V$, a set of edges where and edge $E(v, u)$, starts at *node* v and ends at *node* u
- $\alpha : V \longrightarrow L_V$, is a function to assign nodes labels
- $\beta : E \longrightarrow L_E$, is a function to assign edges labels

In Figure 3, color circles are nodes, and black lines are edges. The set of strings $Ov, DC\ and\ Cvx - Ov$ are edges labels and represent a *spatial relationship* between nodes. Further explanation about spatial relationships can be found in Section 2.3.

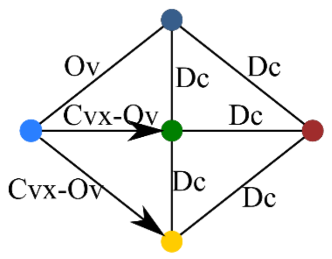


Figure 3 Graph: nodes, edges, and labels (M. Chipofya et al., 2017)

Graph matching involves estimating the configuration similarity by finding a correspondence between edges and nodes of a pair of graphs fulfilling several constraints to find similar substructures on one graph into the other (Conte, Foggia, Sansone, & Vento, 2004). The comparison between graphs is classified into two main approaches, *Exact* to find isomorphic relations or *Inexact* to asses an approximate solution, depending on how elements are paired (Foggia et al., 2014). Exact graph matching is usually restricted to a set of problems and have a binary solution: a match is true or false, whereas *Inexact* or *error-tolerant* matching is capable of handling real-world class distortions and providing an evaluate the level of similarity between two graphs but is more expensive to compute (Cook & Holder, 2006; Emmert-Streib et al., 2016).

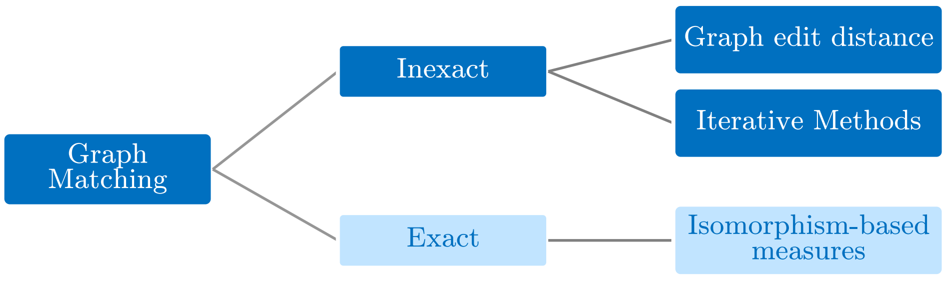


Figure 4 Graph Matching techniques (Conte et al., 2016)

For solving the error-tolerant matching, one of the most used formalizations to the use of the edges' constraints is the *weighted graph matching* in which the graphs are illustrated by the corresponding adjacency or *similarity* matrices (Foggia et al., 2014) . Given two graphs with similarity matrices A and B, the compatibility between two edges (u, v) and (x, y) can be measure by a function:

$$C_{uvxy} = \begin{cases} 0, & if\ A_{uv} = 0\ or\ B_{xy} = 0 \\ c(A_{uv}, B_{xy}), & otherwise \end{cases} \qquad 2.1$$

where $c(.,.)$ is a defined compatibility function. The correspondence solution to this graph matching category includes algorithms designed to compute an approximation of the Graph Edit Distance obtained from node-editing actions (delete, insert) and constraints are still satisfied (Conte et al., 2004), others are based on properties related to the eigenvectors of the adjacency matrix referenced as *Spectral Techniques*, as well as Iterative Methods on the other hand for studying repetitive arrangements derived from the calculus of similarities scores (Cho, Lee, & Lee, 2010; Foggia et al., 2014). These methods evaluate the node's vicinity to assign correspondences during the search, and their application is linked to the nature of the problem. Other approaches include heuristic techniques for combinatorial situations, such as tabu search, which are described in section 2.13 and 3.33.

2.3 QUALITATIVE SPATIAL REPRESENTATION

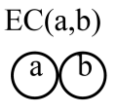
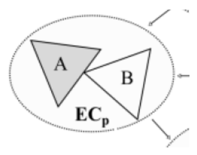
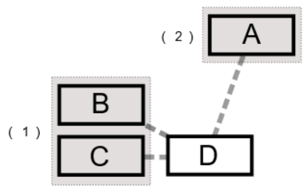
As Sketch Maps does not have a georeferenced system, it is necessary to automatize the analysis of spatial relationships to identify the underlaying correspondence between the elements represented (Wallgrün et al., 2010). Furthermore, the system requires the appropriate constraints design to establish correspondences to the desired dataset, such as a metric map (Malumbo Chipofya et al., 2011). These constraints are derived from the encoding process from physical experiences in which we applied our reasoning in daily activities, generating knowledge to describe the relationships between elements in the surrounding space (Štěpánková, 1992). The *spatial relations* like *adjacency* or *inclusion* for elements such points, lines, or regions are described by *qualitative representations* from the perspective of direction, position, or the physics of space (Jan, Schwering, Schultz, & Chipofya, 2015). Instead of numerical labels to define the structure of the physical world, qualitative representations illustrate our perception from specific conceptual distinctions (Freska, 1991). In order to calculate these representations, different *qualitative spatial calculi* are applied to be organized as constraints in a new graph and constructing a Qualitative Constraint Network (QCN) (M. C. Chipofya et al., 2016). This leads to the idea that finding correspondences between qualitative spatial relationships from a sketch map to a metric map can be done through the match of the equivalent QCN for each map (Malumbo Chipofya et al., 2013).

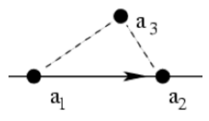
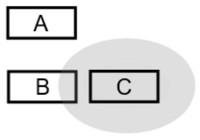
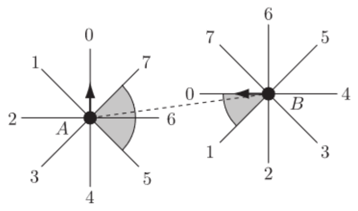
In the following subsections, we detailed the different spatial calculi and QCN structures developed for giving a solution to the problem of finding correspondences between a sketch and a metric map.

2.3.1 Qualitative Spatial Calculi

A *qualitative calculus* is defined as the set of algebraic structures to describe qualitative reasoning between objects which constitute the domain of the calculus (sharing the same type: line, points, or regions) by assigning a *relation* (Malumbo Chipofya et al., 2013). Table 1 displays a subset of the available spatial calculi involved in the graph matching problem for the alignment in sketch maps derived from empirical studies (Malumbo Chipofya et al., 2011; Jan et al., 2018):

Table 1 Spatial Calculi for Qualitative Representations

Calculi	Description	Example
RCC8	Eight topological relations based on the primitive relation $C(x,y)$ (Randell, Cui, & Cohn, 1992)	 **EC** Externally Connected
RCC11-LPC	Eleven topological relations between city blocks based on the dim of the intersection of boundaries (line or point contact) (Jan et al., 2015).	 **ECp** Externally connected by a point
Relative Distance	Three relations based on relative metric minimum distance and clusters into three groups (*near, far, very far*) for polygonal features (Jan et al., 2018)	 Object D is far *near* to the cluster (1) and *far* from the cluster (2)

Calculi	Description	Example
$\mathcal{LR}$	Nine relative orientation relations to spatially express a situation for a starting point a_1, reference point a_2 and a focus point a_3 (Scivos & Nebel, 2005)	 Looking from a_1 to a_2, a_3 is to the **left**
Adjacency	Five relative orientation relations (left_of, right_of, front, back, and crosses). It computes the spatial relation between near-by objects (Jan et al., 2018)	 Object B is **left_of** object C
Region starVars	Relative orientation relations which divide the plane into cone-based regions. With a granularity factor m, the number of total relations is $2 * m + 1$. Helps to describe the orientation of one polygon respecting other (Jan et al., 2018; Lee, Renz, & Wolter, 2013)	 0 1 7 2 A 6 3 5 4 6 7 5 0 4 B 1 3 2 A starVars object A with $m = 8$ and angle of orientation $A_\theta = 90°$

Each one of these calculi is useful for delineating and analyze specific arrangements regarding the world that we perceive in reality and construct structures called *constraint networks* to communicate knowledge from a scene (Ligozat, 2005). The next section contains the details about this structure.

2.3.2 Qualitative Constraint Networks

A Qualitative Constraint Network (QNC) is a complete graph in which the edges are labeled from a *qualitative calculus* (for example, RCC11), which describes the relation shared by the endpoints or nodes (Malumbo Chipofya et al., 2013). For a finite set of nodes N, a set of relations A and $C: N \times N \rightarrow A$ a projection which to each set of nodes (i, j), we assign an element $C(i, j)$ of A called a *constraint* on the edge (i, j). In Figure 5, the nodes or pairs are illustrated in color circles (N) and their corresponding label or constraints ($C(i, j)$) from the RCC8 calculus relation set (A).

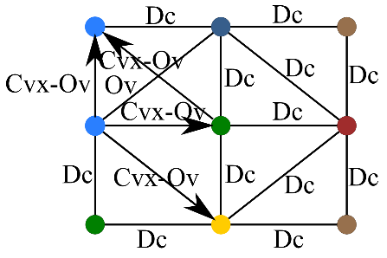


Figure 5 Graph labels and nodes (color) (M. Chipofya et al., 2017)

There are three properties in qualitative reasoning to asses *consistency* in a constraint network. A network is said to be (Ligozat, 2005):

1. *Normalized:* if the node (i, j) labeled by C and the node (j, i) is labeled by $C(i, j)^{-1}$ for all (i, j)
2. *Atomic:* if $C(i, j)$ has only one basic relation for each pair (i, j)
3. *A-Closed:* if for every triplet of nodes (i, j, k) exists $C(i, j); C(j, k) \supseteq C(i, k)$

Consistency is achieved if there is an appropriate structure along with the constraints (Ligozat, 2013). In particular, if in a constraint network every restriction is coherent then, it is said to be closed and stablishes the consistency of a QCN with a spatial calculus A, leading to the exercise of encountering correspondences for a set of qualitative spatial representations as the solution for the QCN matching problem (Malumbo Chipofya et al., 2013).

As it is a high order dimensionality problem, we need more specialized structures to find matches efficiently (M. C. Chipofya et al., 2016). In the following sections, we highlight the use of local compatibility matrices constructed from the qualitative constraint networks.

2.3.3 Local Compatibility Matrix

A Local Compatibility Matrix (LCM) is a case of QCN derived from two graphs qualitative analysis, offering a global representation for the correspondence for a set of pairs during the match search for an input graph (Malumbo Chipofya, 2018).

An LCM states the compatibility between a *specified* pair $(i, i') \in NxN'$ and $(j, j') \in NxN'$ and every other pair. In the matrix, a *row* corresponds to the input node $j \in N$, a *column* the target node $j' \in N'$ and, the *cell*, the largest label common to both edges $l(i,i') \cap l'(j,j')$. Represent the compatibility between every pair requires $|N| \cdot |N'|$ LCMs (M. C. Chipofya et al., 2016).

Properties from LCM are derived from its geometry. The first one is the possibility to sort rows and columns in a way that the cell with the same labels forms rectangular submatrices. Secondly, for non-overlapping and equal labels, these submatrices do not overlap each other (M. C. Chipofya et al., 2016). Extracting information about the local compatibility in this structure requires the computation of two heuristic scores, which is detailed in the System Scoring subsection 3.5.3.

2.4 LINK ANALYSIS

Traditional methods to recover information about a graph structure are focused on encountering a substructure to obtain a set of probabilities distribution (Dehmer, 2008). Finding a solution to the graph matching problem in computing engineering for pattern recognition, for example, has derived methods ranging from the manipulation of the similarity matrix to the redefinition of the graph class to obtain new similarity measures (Cour, Srinivasan, & Shi, 2007; Dehmer, Emmert-Streib, & Kilian, 2006). One approach is spectral techniques developed in computer vision, giving consistent results in identifying the correspondence between features analyzing the compatibility of the geometric constraints with the idea of identifying clusters from highly related items to fulfill an approximate solution contribute some insights to the current design of matching algorithms (Leordeanu & Hebert, 2005).

2.5 REINFORCEMENT LEARNING ALGORITHMS

Diverse techniques for matching a variety of features, including multi-polygons, have been developed in computer science (Bunke & Jiang, 2000). As the search space increases, these techniques need to be able to handle significant inputs of information and offer the possibility to find patterns (Foggia et al., 2014). In this regard, learning algorithms offer a routine in which is possible to improve the performance: it stores the data during the *agent-environment* interaction, maximizing the weight of the backup information with a set of received rewards in a Markov Decision Process, to organize and structure the search and make appropriate decisions, based on the environment arrangement (Sutton & Barto, 1999).

One of the keys configurations in reinforcement learning algorithms is the pertinent generation of the *action-value* and the *state-value* functions. By correctly identifying the conditions for selecting a feature in the case of the correspondence problem, the optimal solution computation time may improve, learning to associate potential aggregation with profitable regions of the search space to mitigate the adverse effects of an exponential expansion of the search space (Chipofya,2016). If the agent experiences future lower rewards, it returns to a past *state* in which a better next step or selection exists. Reinforcement learning techniques are an approach to make the best decision from the exploration and identification of situations and their consequences (Sutton & Barto, 1999).

For any Reinforcement Learning problem,

t are the steps in which the environment receives a state

s_t is the environment state at the step t such as $s_t \in S$ where S are all the possible states

The agent then selects an action accordingly to its current state, a_t is an action such as $a_t \in A(s_t)$ where $A(s_t)$ are all the actions available in the state s_t. As a result of this action, the agent receives a numerical reward, and the agent advance to a new state. By doing so, the agent is pursuing a mapping, formally called a policy, from states to probabilities of selecting each possible action:

π_t Is a policy, a mapping from each state $s \in S$ and action $a \in A(s)$ to the probability $\pi(s, a)$ of selecting action a when the agent is in the state s

From this, by following the policy in a specific state, the expected *state-value* is obtained in

$$V^{\pi}(s) = E_{\pi}\{\textstyle\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \, | s_t = s\} \qquad 2.2$$

Furthermore, by following a policy starting in a specific state and taking a specific action, the expected *action-value* is obtained in

$$Q^{\pi}(s,a) = E_{\pi}\{\textstyle\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \,|s_t = s, a_t = a\} \qquad 2.3$$

The goal is to have a good enough policy that maximizes the reward reflected in the V and Q values. Producing an optimal approximation implies to select the best value available, by backing up and comparing these results. The concept of Dynamic Learning is visible in this approach: by iteratively evaluate the best possible decision combination (policy) in a value function, it is possible to find the best solution to the selection process.

CHAPTER 3 METHODOLOGY

This chapter focuses on the methods implemented to answers the research questions. First, a description of the implementation workflow is presented in section 3.1 to illustrate the connection between each module. The system setup is described in section 3.2, followed by the description of the data used to test the searching algorithms. Next, in Section 3.4, the scoring system components are presented, giving details on how they are calculated. Finally, section 3.5 describes the design and pseudocode for the Tabu and SARSA searching algorithms, respectively.

3.1 IMPLEMENTATION WORKFLOW

The process of aligning one feature to another between a sketch and a metric map requires the integration of the new implementation with the SmartSkeMa functionalities. In the following sections, we refer to the developed solution as the Dynamic Programming for Sketch Maps, *DPSM*. First, to recognize the drawing, the metric and the sketch maps are uploaded as inputs in the of the SmartSkeMa interface to be displayed and processed in the backend (1. Input Processing). Following the Qualitative Segmentation, a computer vision process that identifies the features in the sketch map (for more information about this process, review Murcia, 2018), the data obtained from the vectorization is used by the Qualitative Representation module to derived the relation set values and return the Qualitative Set and the Similarity Matrix, two inputs used in *DPSM* for assessing the compatibility and the similarity between features (2. Qualitative Analysis). Once the data is received from the mentioned modules, three different scores *Link Analysis, Spectral Solution*, and *Local Compatibility heuristics,* are calculated and provided (3. Score System). Next, the searching algorithms start the alignment process with the scores as arguments (4. Searching Algorithms). A more detailed review of the different modules is:

- **Input processing:** Using the SmartSkeMa's project interface, the sketch and metric maps are processed. SmartSkeMa will execute the Qualitative Segmentation (recognize the sketch maps features).

- **Qualitative Analysis:** once SmartSkeMa vectorizes the image, it will run the process of Qualitative Representation, giving as outputs the Similarity Matrix and the Qualitative Constraints datasets. In the DPSM implementation, we evaluate the compatibility between candidates' pair and temporary solutions.
- **Scoring System:** this module is useful to calculate different scores for each candidate pair as a criterion to evaluate if it will be added to the current solution. One of the tasks is to recreate the Local Compatibility Matrix for each candidate pair and the current solution to calculate the LCM heuristic score. It is also used to calculate the Link Analysis score for each pair considered based on the similarity matrix. Finally, by using the LA score, it will calculate the Spectral Solution to be considered as initial solutions for the algorithms as it will show highly connected pairs inside the search space.
- **Searching algorithms:** two different algorithms are implemented to give a solution to the alignment problem: a Tabu search, from the non-learning algorithms, and a SARSA algorithm from the reinforcement learning algorithms. Both are fed with the output generated in the previously mentioned modules given. As a result, a list of sketch and metric maps features to be displayed in the SmartSkeMa interface.

Finally, the searching results will be evaluated by the performance (execution time in seconds), precision, and recall. In this section, the concepts will be explained in more detail. The implementation diagram is detailed in Figure 6:

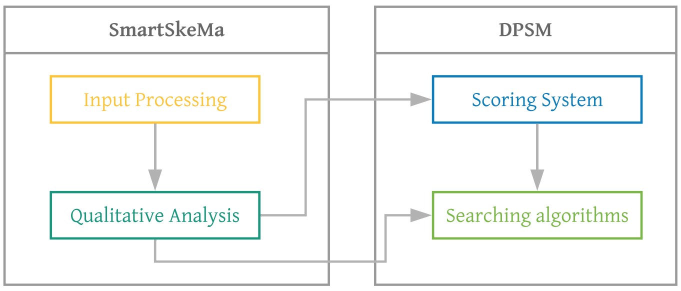


Figure 6 Framework implementation workflow

For each major component, every module is responsible for a set of process and outputs that are going to be used in future functionalities; this workflow is shown in Figure 7 with gray dotted lines for the modules used from the SmartSkeMa framework:

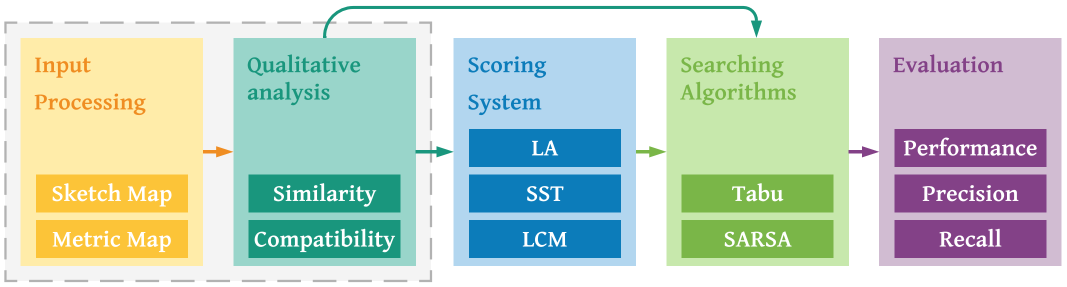


Figure 7 Thesis implementation workflow

3.2 SYSTEM SETUP

For workflow implementation, it is necessary to integrate different libraries and modules in a unique environment. For the system setup, the following libraries were used from and in the SmartSkeMa framework.

3.2.1 Python

The proposed methodology is developed in **Python**. *Python* is an object-oriented programming language with high-level data structures as it offers diverse standard libraries from string processing to system interfaces, some of which are specially design and optimized to handle large datasets (The Python Software Foundation, 2009).

The main Python libraries implemented are:

- *Numpy (v. 1.18.1):* package dedicated to scientific computing with Python offering tools for manipulating N-dimensional array datasets. We use the matrix tools to store, manipulate and process data (NumPyCommunity, 2020)
- *Intertools (v. 2.3):* this module offers fast and memory-efficient tools to iterate through data. As in our research, it is needed to search in large datasets. This module helps to optimize the process in between during the solution exploration (The Python Software Foundation, 2003b).

- *Collections (v. 2.4):* module implementing specially designed container datatypes as alternatives to the Python's standard built-in ones, with a high-performance outcome in our case for storing and manipulating the data in the implementation (The Python Software Foundation, 2003a)
- *OpenAI Gym:* a library with a collection of test problems called *environments* to implement reinforcement-learning algorithms with a shared interface (Brockman et al., 2016)

3.2.2 SmartSkeMa framework

The Smart Sketch Map system (SmartSkeMa) is an application to record sketch-based information regarding land tenure in the frame of peri-urban and rural territories displaying an integrated vision of the end user's sketch map and a cartographic dataset (M Chipofya, Jan, Schultz, & Schwering, 2017). An overview of the interface is displayed in Figure 8 with the input sketch map in the left and in the right side, the corresponding metric map.

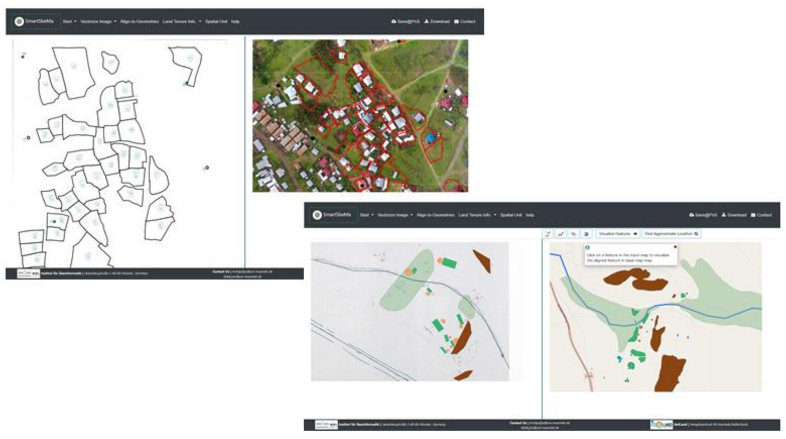


Figure 8 SmartSkeMa interface

From the SmartSkeMa project, the main modules used are:

- *Sketch recognition:* for identifying distinctive elements in the sketch maps, for instance, water bodies, houses or mountains, the module processes shapes, and features' representations by using a symbol recognizer to extract visual representations and transforming them into vector geometries (see Figure 9) to be stored in the system, using supervised learning techniques, polygonal clustering methods (Ng & Han, 2002) and image recognition methods (K. Broelemann, 2011; Klaus Broelemann & Jiang, 2013)

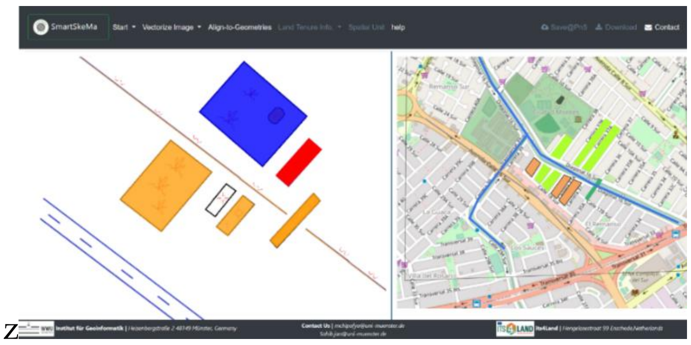


Figure 9 SmartSkeMa vectorization

Through the interface, we provide the sketch and the metric map files and run the *Automatic Vectorization* process, and additionally, we provide more vectorized features using the *Geometry Editor* functionality. The interface described is displayed in Figure 10:

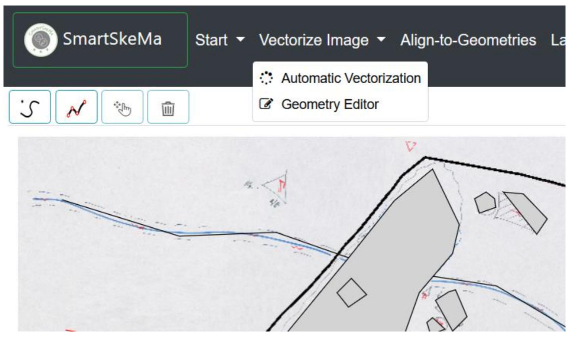


Figure 10 SmartSkeMa Geometry Editor user interface

- *Qualitative Representation:* In the sketch to map alignment problem, every item is defined as a node inside the qualitative map with a designated class and the corresponding attribute values to identify them. Spatial relations are used to describe the location of each item in the qualitative spatial representation, becoming *labels* between each *node* in a *graph* matching model, and only a set of qualitative calculus are combined with stabilizing the distortions captured from the sketch map (M Chipofya et al., 2017).

As a result, the module generates the corresponding *Qualified Map* for the sketch and the metric maps and the *Similarity Matrices*. Both datasets are organized based on the *candidate pairs*; these are each possible combination between an element in the sketch map that may correspond to another element in the metric map. The *Qualified Map* dataset describes the labels between each node in the graph representation of the sketch to map association per relation set. On the other hand, the *Similarity Matrix* contains binary information about the compatibility between each label for every candidate pair.

3.2.3 Metric map generation

For the generation of the metric map's dataset, the software QGIS in the version *3.10 A Coruña* is used to digitalize the areas' features and export them as a geoJSON file. QGIS is an open-source and multiplatform Geographic Information System (GIS) application supporting raster, vector, and database operations and functionalities for managing geographical information (QGISORG, 2002).

3.2.4 Execution environment

All the procedures were executed on an Intel Core i7-75002U CPU at 2.70GHz, and 8GB DDR4 memory card with a 19GB dedicated virtual memory. A set of environments with different Python configurations are created through *Anaconda*, a scalable data science multi-platform environment manager for packages and Python distributions, with an extensive collection of open source modules to find, access and share (Anaconda, 2020). An Anaconda environment executes a Python version 3.6.4 configuration along with the packages required to run the SmartSkeMa framework. This version of Python is selected according to TensorFlow's version requirements for the Qualitative Segmentation Module.

3.3 INPUT DATASETS

Sketch mapping is a drawing exercise on a large piece of paper that allows recreating a global image of the people's spatial distribution of their territory (Štěpánková, 1992) The following sketch maps are spatial representations with different complexity levels to test the proposed algorithms. All of them have been generated by humans on different platforms, with two of them with the same objective: describe an area of interest according to the mental image of a space previously experienced. In the following subsections, a detailed description of each one is provided.

The common relevant attributes in these representations include:

- *smart_skema_type:* type of feature according to the domain model implemented in the SmartSkeMa framework, derived from the workshops carried in the Maasai community in Kenya for the its4land project (Karamesouti et al., 2018; Murcia, 2018). The features' type catalog is detailed in Table 2.
- *name: a* descriptive label for each feature for identifying purposes
- *id:* feature unique identifier for different processes inside SmartSkeMa and the DPSM implementation.

Table 2 SmartSkeMa feature types

Feature type	**Description**
beacon	An object for specifying land boundary
boma	A small place where people rest
boundary	Clear delimitation of an area
house	Standard family living unit
marsh	Large wetland with plants. Associated to green areas in the experiment
mountain	Represents a single mountain or chain of mountains
olopololi	Area for agricultural activities. Associated to bridges in the experiment
oltinka	Water collection site
river	Natural or human-made water currents
road	Human-made access with or without pavement surface
school	Building with educational purposes
tree	An area containing one or more trees
water_tank	Water storing area for a collective usage

3.3.1 Artificial dataset

The artificial dataset is a set of different elements with a random distribution. In Figure 11, the resulting sketch map is displayed. On the left side is the sketch map representation, and on the right side, the objective metric map to align.

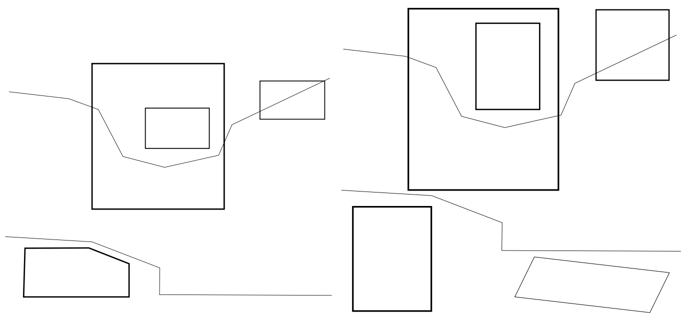


Figure 11 Artificial dataset SVG

Each one of the map representations contains the features described in Table 3:

Table 3 Artificial dataset features

Map type	Number of features	Type of features
Sketch	6	(2) Marsh (1) River (1) Olopololi (1) Road (2) Mountain
Metric	7	(2) Marsh (1) River (1) Olopololi (1) Road (2) Mountain

3.3.2 El Remanso

El Remanso is a small neighborhood located in Bogotá, Colombia, in a residential area between the Fucha river (blue line) and the Primera de Mayo Avenue (yellow line), as displayed in Figure 12. The community enjoys green areas around the river, such as the Ciudad Montes Park, which has a small lake (blue circle). People from the southwest side of the river can cross using a bridge (purple line). Bogotá is known for the mountain chain in the east called Eastern Hills, as they are visible from most of the citizens and serves as an essential spatial reference element in the landscape (Pavony, 2000; Robson, van Kerkhoff, & Cork, 2019)

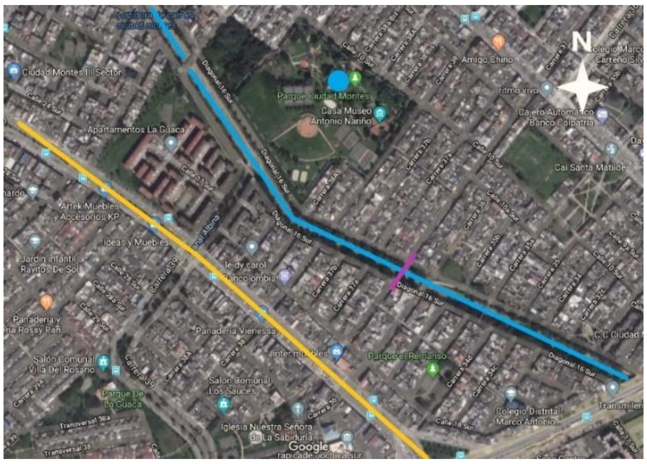


Figure 12 El Remanso neighborhood (Google Maps view)

The sketch map represents the mental image of the main elements recalled from the neighborhood, as shown in Figure 13:

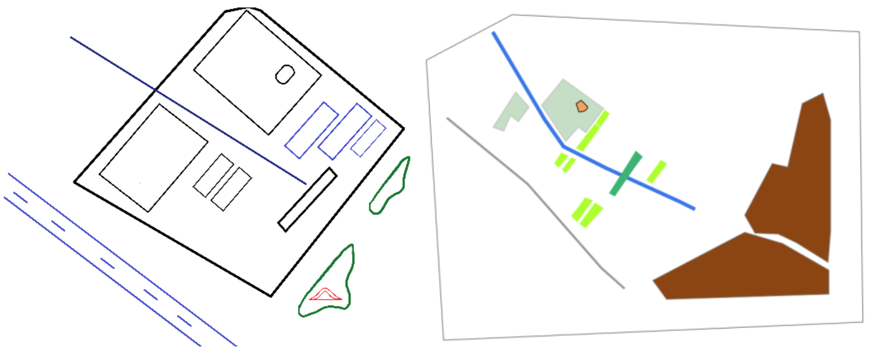


Figure 13 El Remanso: sketch (left) and metric (right) maps

The metric map geoJSON file is created in *QGIS A Coruña,* and the attributes are filled according to the SmartSkeMa guidelines. Each one of the map representations contains the features described in Table 4:

Table 4 El Remanso dataset features

Map type	Number of features	Type of features
Sketch	13	(5) House (2) Marsh (1) Boma (1) Olopololi (2) Mountain (1) River (1) Road
Metric	15	(7) House (2) Marsh (1) Boma (1) Olopololi (2) Mountain (1) River (1) Road

3.3.3 Mailua Ranch

The Mailua Ranch is a sketch map data set collected in the Maasai community located in Southern Kenya, in which the SmartSkeMa project participates with other partner universities to provide tools in the land rights for the area residents. The sketch map in Figure 14 was created by individuals from the Maasai community in one of the field studies where additionally the domain model was generated for the spatial components described including classes for environmental characteristics, social units, activities, shapes, housing, and farming (Karamesouti et al., 2018).

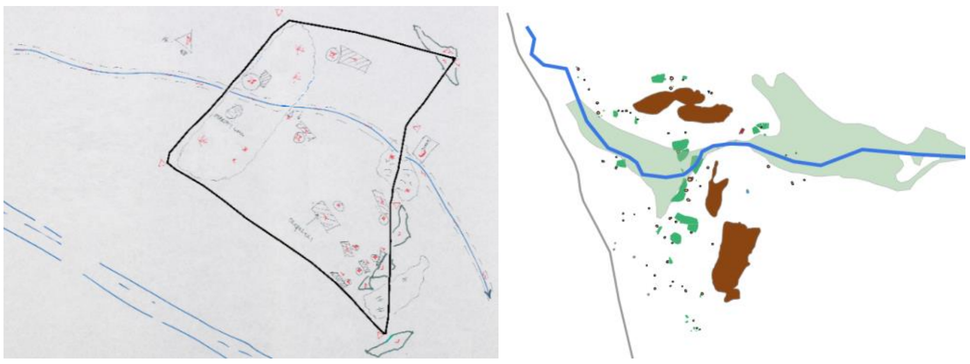


Figure 14 Mailua Ranch: sketch (left) and metric (right) maps

The sketch map object representations contain the features described in Table 5:

Table 5 Mailua Ranch dataset features

Map type	Number of features	Type of features
Sketch (geometry editor)	16	(1) School (1) River (2) Road (3) Mountain (1) Marsh (3) Boma (5) Olopololi
Sketch (vectorization)	31	(1) School (1) River (2) Road (4) Mountain (5) Marsh (8) Boma (11) Olopololi
Metric	106	(1) School (1) River (2) Road (3) Mountain (3) Marsh (n) Boma (n) Olopololi

3.4 QUALITATIVE ANALYSIS

The sketch to map features alignment is approached as a graph matching problem in which every map feature is defined as a node and each relation label as an edge, as described in section 3.2.2. In the SmartSkeMa *Input Processing*, the system generates the Similarity matrices and Qualitative Constraint Map (QCM) and stores them to be used in the Qualitative analysis module, responsible for providing the compatibility and the similarity evaluation between two candidate pairs during the execution of the searching algorithms.

A *candidate pair p* is a set of features (i, i') where i $\in X$ and $i' \in Y$, in this case, with X representing the sketch map and Y the metric map to align.

The main tasks of this module are to return values for:

- *Similarity*: the similarity between the two pairs, p_1 and p_2 is recovered from the similarity matrix $N \times N'$. Moreover, it is evaluated as:

$$similarity(p_1, p_2) = \begin{cases} True, & if\ N \times N'_{p_1, p_2} = 1 \\ False, & if\ N \times N'_{p_1, p_2} = 0 \end{cases} \quad 3.1$$

- *Type Compatibility*: the type compatibility for a pair (i, i') is evaluated from the QCM feature type attributes QM as:

$$type\ compatibility(i, i') = \begin{cases} True, & if\ QM(i) = QM(i') \\ False, & otherwise \end{cases} \quad 3.2$$

- *Candidate-Solution Compatibility:* given a current solution m, for a candidate pair p' the compatibility with m is:

$$c(m, p') = \begin{cases} True, & if\ \forall\ p \in m\ \ similarity(p, p') = 1 \\ False, & otherwise \end{cases} \quad 3.3$$

The definition of these values helps to filter the search space during the selection of candidate pairs for a current solution to the ones who add value to the final solution.

3.5 SCORE SYSTEM

The evaluation of a pair is a critical process in the alignment problem as the decision of the next most fitting step must contemplate the impact of the extension of every QCN, each relation set, and additional considerations (Chipofya, 2013). In this thesis, the additional considerations are based on the graph matching solutions for discrete problems in a closed graph, which state the influence of the vicinity configuration (R. Battiti & Protasi, 2001). From the Similarity Matrix, a Link Analysis is used to extract the ranking scores of each node in terms of their connectivity, taking these results to extract an initial solution with the Spectral Solution Technique studying, in this case, the clustering behavior of the nodes. Finally, from the neighborhood properties of the LCM, two heuristic scores are calculated.

3.5.1 Page Rank

PageRank is an algorithm developed to *ranking* a node according to the number of links in a web graph, by assigning a score between 0 and 1; during the graph exploration, some nodes are more visited than others, creating a network in which profoundly explored nodes share a high number of connections in between. (Ceri et al., 2013; Page, Brin, Motwani, & Winograd, 1998a). The result is a *distribution probability vector* or also called the *left-eigenvector*, representing the ranking score for the candidate pairs in the Similarity Matrix A.

Consider a web graph G in which pages are represented by nodes N. Let u be a web page (node) pointing to a set of pages (nodes) F_u and in the same way B_u the set of nodes pointing to u. Let $L_u = |F_u|$ be the number of links (edges) from u and c to be a normalization factor. The equation gives the simplified version of PageRank ranking value R:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{L_v} \qquad 3.4$$

In order to calculate the corresponding PageRank score, the following variables and procedures are addressed:

- *Teleport operation:* if N is the total number of nodes in the web graph G, the operation to move from one to another happens with a probability of $\frac{1}{N}$. A teleport rate with probability $0 < \alpha < 1$ is defined to avoid looping in nodes with low compatibility and encourage exploration.
- *Initial probability distribution vector:* base vector for the *distribution probability vector* as it represents each node value procured by dividing the sum of the number of nodes connected to it by the total number of features connected. In the case of the sketch to map alignment, the number of nodes connected is the ones with value 1 in the Similarity Matrix, as it represents the compatibility of each one of the features in the search space.
- *Transition Probability Matrix P:* Consider a graph G with a set of nodes *N= {A, B, C}*. The matrix *P* represents the distributed probability of moving from one node to another, as seen in Figure 15:

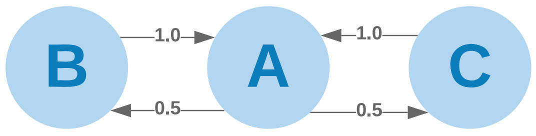


Figure 15 Transition probabilities for a graph G (Ceri et al., 2013)

A row represents a candidate pair in the Similarity Matrix, and it is divided by the number of compatible features in that row. For the resulting base matrix P, a *Teleport Distribution* $(1-\alpha)$ and *Teleport Variation* (α/N) is applied.

- *Power iteration:* the method implemented to calculate the left-eigenvector and the corresponding largest eigenvalue of a matrix, named the Distribution Probability Vector, the ranking score. Some of the advantages of this method include that it does not affect the *transition probability matrix* P, can handle large sizes of data, and it returns the values of interest in less computational and complexity expenses.

The *Link Analysis* (PageRank) score computation in DPSM is executed with the *Similarity Matrix*, calculated from the input sketch and metric maps, as an argument, and the system retrieves the ranking scores for all candidate pairs. The link analysis results are used in the Spectral Score Technique (SST) as an argument to calculate an initial solution and, in the Tabu Search, to make a move in the selection process. In Figure 16, the PageRank implementation for the sketch to map aligning problem is detailed.

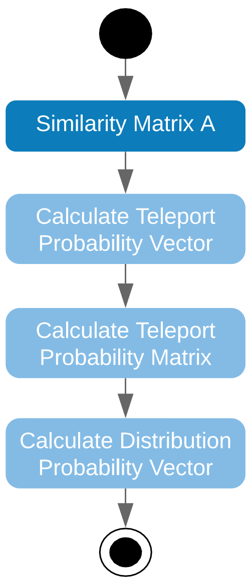

Similarity Matrix A
Calculate Teleport Probability Vector
Calculate Teleport Probability Matrix
Calculate Distribution Probability Vector

Figure 16 Link Analysis evaluation diagram

3.5.2 Spectral Solution Technique

The Spectral Solution Technique is an algorithm from Leordeanu and Hebert's research, able to find secure correspondences between a pair set of features as *nodes* in the graph matching problem, by calculating the eigenvector of a graph matrix *M* and processing these scores to get a collection of highly linked assignments. In this algorithm, selected features are highly related and expose high links scores showing a clustered behavior among them. On the other hand, low related features do not show any links rates, or if they appear, they show a considerable distance concerning the central cluster (Leordeanu & Hebert, 2005).

Consider a graph *G* with a set of nodes *N* represented by the matrix *A*. Initially the similarity matrix is constructed, followed by the definition of environment variables: *L* as the number of nodes, *x* as the elements of the row in the iteration, and x^* the maximum eigenvalue or affinity scores for the Similarity Matrix *M*. As the SST addresses the use of maximum eigenvalue, it makes use of an algorithm that pursues the identification of links between large amounts of objects connected. One approach is Google's algorithm, PageRank, which calculates a feature relevance inside a network according to the number of links shared with other features (Page, Brin, Motwani, & Winograd, 1998b). The algorithm will reject all the objects in the iteration with a lower value and a corresponding label in conflict with x^* and collect the high scored and compatibles ones as long there are features left to analyze in *L*. Finally, *x* will contain the pairs candidates with the highest confidence of being correct assignment. The algorithm executes the steps described in Figure 17. In general, in the graph matching problem, the SST candidate pairs serve as a start point for the exploration in the search space. For the sketch to map alignment, these selected features represent the pairs candidates with the most number connections or relation labels inside the sketch-metric map graphs being good candidates for initializing the search. Moreover, according to the Link Analysis theory and Leordeanu findings, from the analysis of the candidate pair's vicinity and identifying the existence of robust correlated features, the definition of the first steps during the search task make the results more profitable than resume from a point with no information available (R. Battiti & Protasi, 2001).

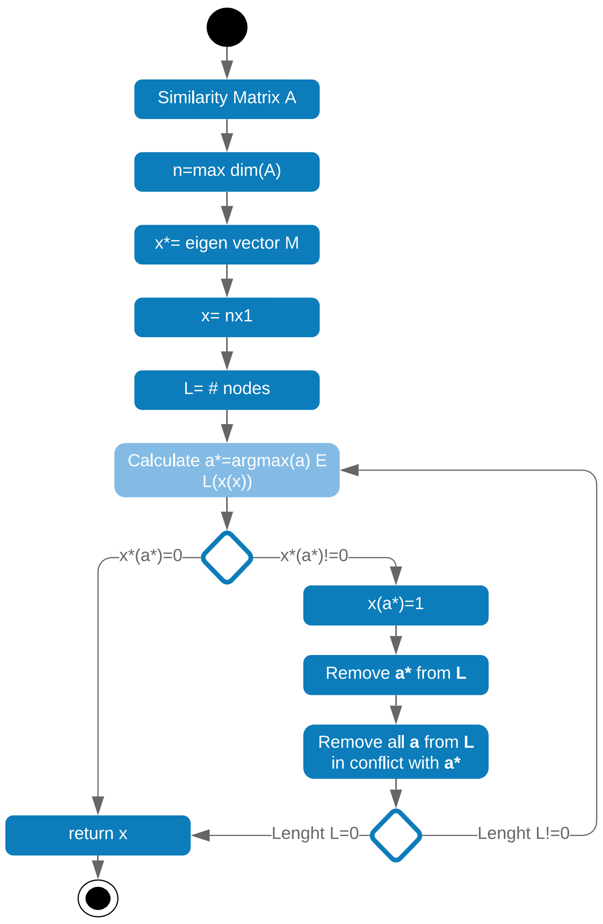


Figure 17 Diagram for the Spectral Solution Technique algorithm

3.5.3 Heuristic scores from LCM

The third score component is based on the properties of the Local Compatibility Matrices described in section 2.3.3, especially the *non-overlapping labels* property in which matrix cell with the same label generate a non-overlapping square submatrix inside the LCM from which two heuristic scores are derived (Malumbo Chipofya et al., 2013):

Consider a candidate pair (i, i') with LCM $\mathcal{L}_{(i,i')}$and its corresponding submatrices denoted by $\mathcal{L}_{(i,i'),R}$ with rows $rows(\mathcal{L}_{(i,i'),R})$ and columns $cols(\mathcal{L}_{(i,i'),R})$, where R is a label $R \subseteq \boldsymbol{R}$. The first observation of this configuration is the possibility of identifying a set of submatrices inside $\mathcal{L}_{(i,i')}$ furthermore considering the square submatrix property, get the minimum submatrix dimension which indicates the highest contribution of each $\mathcal{L}_{(i,i'),R}$ into the extension of the current candidate pairs match m in the future: the highest the total sum of the min dimension of the submatrices in the LCM, higher the chances to find in the future more compatible candidates in the solution as indicated in the equation:

$$eval_{i,j|m} = \sum_{R \subseteq \boldsymbol{R}} \min\left(\dim\left(\mathcal{L}_{(i,i')}(R)\right)\right) \qquad 3.5$$

The result is a greedy selection of candidate pairs, as the selection follows the paths labeled as useful in the first consideration. The second heuristic *H2* complements the first heuristic *H1* by providing a peak in the estimation of a good pair in the solution evaluating the impact of the current pair (i, i') into future solutions by ordering the candidate pairs in ascending order of *H1* and considering the most significant feature, $head_i$ as the *possible* solution that contains the node (i) (M. C. Chipofya et al., 2016) as described in the equation:

$$count_k(m) = \left|\left\{i \in N \middle| k \leq e_{head_i|m}\right\}\right| \qquad 3.6$$

In the DPSM implementation, the first heuristic is calculated by recovering the LCM from the Qualitative Constraint Map (QCM) for a set of candidate pairs. The result is a batch of scores indicating the value of *H1* per each relation set identified from the QCM and finally summarizing them to get a global score. Secondly, the heuristic *H2* is updated for the input map (sketch map) and extended by the SmartSkeMa framework, merging the implementation of the first heuristic. The procedure happens as the candidate pairs are evaluated during the calculus of *H1,* maintaining an updated score structure as the search is executed, adding new features into the solution. Figure 18 describes the implementation for generating the LCM and calculating the *H1* and *H2* scores:

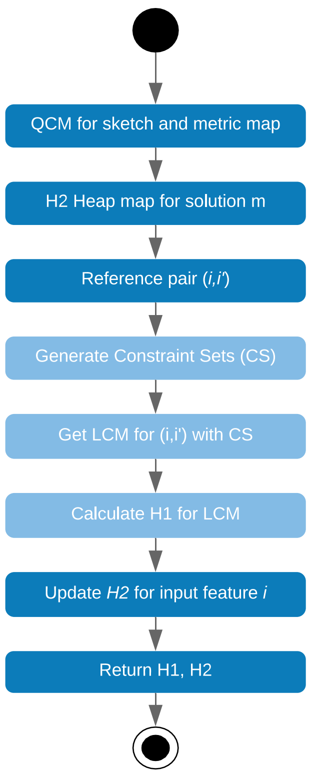

QCM for sketch and metric map
H2 Heap map for solution m
Reference pair (*i*,*i'*)
Generate Constraint Sets (CS)
Get LCM for (i,i') with CS
Calculate H1 for LCM
Update *H2* for input feature *i*
Return H1, H2

Figure 18 Heuristic scores calculation workflow

3.6 SEARCHING ALGORITHMS

The aligning of a sketch map feature to a feature in a metric map is the examination of a large set of options that comply with specific characteristics to be a good match. We explore all the options on the metric map to find which one is the *most like* to match a specific feature in the sketch map. Diverse techniques from non-learning and learning algorithms have arisen from research. In this thesis, we implement two different search algorithms, with different approaches, advantages, and configurations: a Tabu Search and a SARSA, an incremental dynamic programming algorithm to solve reinforcement learning problems (Saad, 2011).

3.6.1 Tabu Search

Tabu search approach is to solve combinatorial optimization problems like the ones in graph theory by using a list of banned or *taboo* moves obtained from a number of iterations in a local search to construct a final solution (Roberto Battiti & Tecchiolli, 1994; Glover, 1989a).

For the configuration of the Tabu algorithm, the main arguments are the search space, the local search space or neighborhood, the list of banned moves, and the criteria to establish whether they belong to the current solution or need to be penalized at each iteration. An overview of the general workflow is shown in Figure 19:

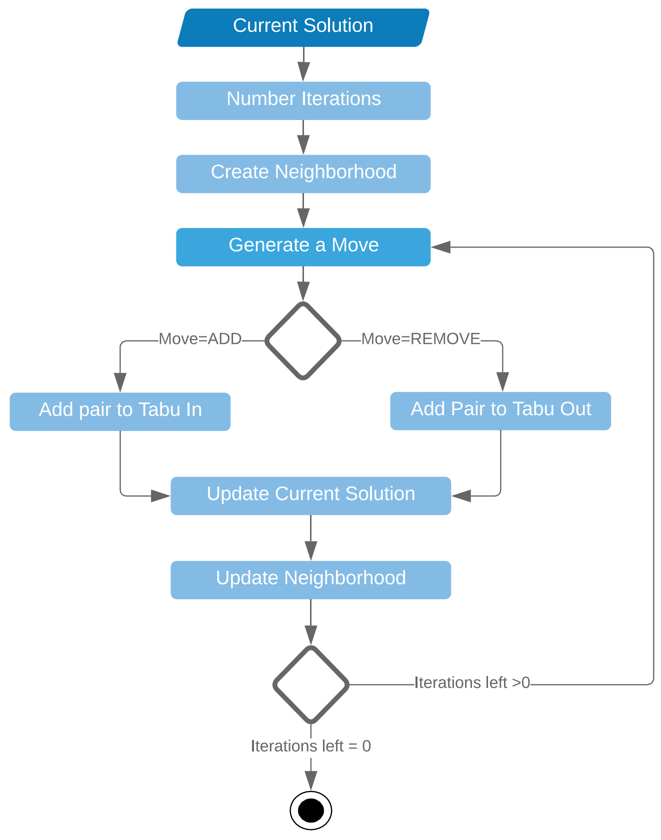


Figure 19 Tabu Search workflow diagram

In the context of the sketch to metric map alignment, the overview for each of these aspects and their processing is as follows:

- *Search Space:* the space of every possible item that can be contemplated as part of the final solution during the search (Gendreau & Potvin, 2005). For the interest of this study, the search space is all possible combinations m composed only by compatible candidate pairs $P = (i, i')$ where i and i' represent a feature from the sketch and metric map.

- *Current solution:* denoted by *S*. The solution used as the initial one is the output from the Spectral Solution Technique.
- *Neighborhood:* the set of available pairs to add to the current solution. For each iteration, a modification or *move* is applied to the solution *S* to add or remove a pair. The result is a collection of available compatible pairs called *neighboring solutions*, a subset of the search space. The evaluation of items belonging to the neighboring solutions is done using the functions created in the *Qualitative Analysis* module. Each new pair added to the current solution *S* during the Tabu Search is compared to every item in the search space *S* to check their compatibility. If old items are not compatible with the most recently added one, they are removed. In the same way, if items from the search space are compatible with the recently added one and with the remaining items, the neighborhood is updated with new available moves.
- *Move:* for each iteration, the algorithm performs a modification to the currently available solution considering all potential actions. For the current implementation, two actions are possible: ADD or REMOVE. The criteria for choosing one or another depends on the evaluation of the neighborhood explained in the following points. The dynamics of a move during the search are displayed in **Algorithm 1**.

Algorithm 1. Tabu Search

```
input: $S_0$, number iterations iter, LA, QSM, QMM, metric_size
output: $S$
// initialize
1     Set S=initial solution $S_0$;
2     Set tabu_in list;
3     Set tabu_out list;
4     while iterations
5         Update available moves
6         Select best non-tabu available move
7         if move is ADD
8             Insert move into S
9             Insert move into tabu_out
10        else move is REMOVE
11            Remove move from S
12            Insert move into tabu_in
13    return best matching result S
```

- *Best non-tabu available move (best_move):* for each iteration, the item with the highest LA score in the neighborhood is selected as the best candidate to be considered in the current solution S.
- *ADD pair:* the search next action is said to be ADD if the *best_move* does not exist in the current solution. The new pair is evaluated using the *Qualitative Analysis* module. For each item in the neighborhood, it assesses the *Candidate-Solution* and *Types Compatibility* values. If both values are positive, the pair is added and labeled as *best_add*.
- *REMOVE pair:* the search next action is said to be REMOVE if *best_move* already exists in S or there are not useful items to be added in the current solution S, with *not useful* meaning a candidate pair that is incompatible with one or more features in S. The procedure is to discard the item with the lowest LA score in S, named *best_remove.*

After executing an ADD or REMOVE move, in both cases, the output element, *best_add* or *best_remove,* is appended to a tabu list. In this implementation, two lists are created: *tabu_in* and *tabu_out.*

- *Tabu lists update:* These are managed by a *FIFO* (First-In, First-Out) method; each time a new element is added to the bottom of the list, the first added element on the list is removed (Glover, 1986). The *tabu_in* list manages the items that we discarded from S and *tabu_out* list the ones we join to the solution. At the beginning of each iteration, the available moves are updated additionally by removing the items in the tabu lists. Instead of using a unique list, we implemented two list, this with the objective of encouraging exploration but on the other hand to not over consider useful elements into the solution, this approach seems to have an advantage in terms of the activity of each list in the algorithm assuring no duplicated solutions while considering candidates inside the solution S (Glover, 1986, 1989b). The size of the tabu lists is fixed to 25% of the size of the current solution.

The algorithm search is executed, and for a given number of iterations, it explores a set of solutions, adjusts the initial solution S by adding or removing pairs from a neighborhood $N(s)$ of s, appearing according to the compatibility to a new solution S'. (Glover, Taillard, & Taillard, 1993). Finally, the best solution is returned with a set of compatible pairs with a size at least equal or more significant than the initial solution.

3.6.2 SARSA

SARSA learns an optimal *action-value* function Q^* from experience gained by an agent while interacting with an environment in an iterative manner in a set of episodes by regularly calculating the value of each *state-action* $Q(s,a)$ (Saad, 2011; Sutton & Barto, 1999). In Figure 20, the dynamics of the SARSA algorithm are described:

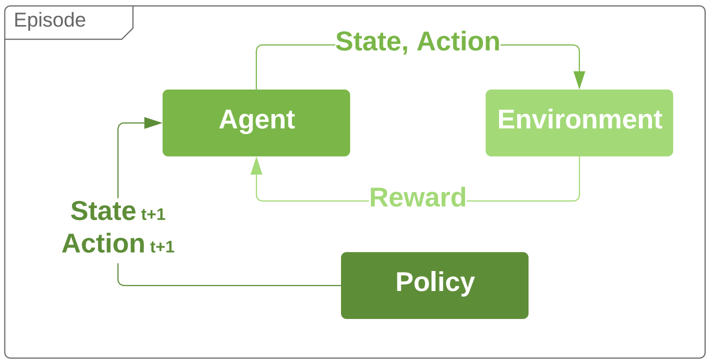


Figure 20 SARSA dynamic

On every step into the environment, the value of the *state-action* pair (s,a) in a step t, is updated according to the received *reward* at step $t+1$and the following selected *state-action* pair (s',a') with a probability ε alternately to selecting it at random (Sutton & Barto, 1999), using a discounted rate α (Ratitch & Precup, 2015) to encourage the exploration and avoid cycling behaviors during the search in contrast with the tabu scheme based on a fixed list size, that is not strict and, therefore, the possibility of cycles remains (Roberto Battiti & Tecchiolli, 1994). The updating of the *action-value* function is given by:

$$Q(s_t,a_t) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \qquad 3.7$$

The equation leads to the progression of (s,a) to (s',a') by using the values $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ in a sequence of **S**tate, **A**ction, **R**eward, **S**tate, **A**ction. The algorithm learns during the episode that some policies are weak and switch to another one. For the specific case of the sketch to metric map alignment, we define the following attributes for the algorithm set up:

- *Step:* a step is an iteration in which the agent will take *action* for the current *state*
- *Action*: an action in the graph matching problem is each one of the candidates' pairs the search space as they are the options available to move into the environment. Each action returns a *reward* if they are added to the current *state* or solution. An action is said to be the *next action* when a new candidate pair is selected among the available pairs from the calculus of the action probabilities. The *action probabilities* are returned by the *policy* and represent the probability for each action to be chosen in the next step. The values are calculated based on the LCM scores modified by ε.
- *State*: a state in the graph matching problem is every set of compatible candidate pairs. Every state is different, has a different accumulated reward at a step t, and represents a possible solution for the search. At every step into the environment, a *new action* is executed (this means a pair is added to the solution) and creates a *new state* (a new solution is generated from the previous solution plus the new pair). In the SARSA algorithm implementation, the initial *state* is selected randomly from the *action space* or candidate pairs returned by SST. The *final state* is found when there are no more items compatible with the current solutions, this means, a peak has been found and the algorithm "walked" into a dead end while connecting the candidate pairs.
- *Reward*: is a numerical value that the algorithm maximizes in the search. At each step, a reward is calculated based on the effects of future action on the current state. For the sketch to map alignment, the reward is based on the score of the candidate pair from the LCM calculus.
- *Discount rate* γ: determinates the current value of the next rewards by considering a reward earned in the forthcoming k time steps with γ^{k-1}. The discount rate $\gamma \in [0,1]$ is a constant, the closest to 0 it will maximize the most recent rewards; the closest to 1 the later rewards will have more weight. From empirical results, a discounted rate $\gamma = 0.9$ is used in most of the cases to avoid getting into a greedy algorithm, that is, maximizing only immediate rewards (Sutton & Barto, 1999, p. 55)

- *Step-size α:* fixes the updating pace for the values. The step-size parameter $\alpha \in [0,1]$ is a constant, as the reward probabilities do not change over time (Sutton & Barto, 2018, p. 32)
- *Probability ϵ:* to avoid selecting the best option while evaluating the available actions that is a greedy action, they are affected by a probability $\varepsilon \in [0,1]$, resulting in a random selection. For all the actions $a \in A$, the low scored actions are given a probability of selection equal to $\frac{\varepsilon}{|A(s)|}$ and for the high scored actions, in the case of this implementation only to the best possible action from the LCM scores, the probability $1 - \varepsilon + \frac{\varepsilon}{|A(s)|}$ is given.

For the current implementation, the workload is divided into two modules: the SARSA Main, which controls the *action-value* function updates, and the SARSA Environment, which contains the logic behind the policy evaluation.

- *Episode:* An episode consists of an alternating sequence of states and state-action pairs (Sutton & Barto, 1999). The number of episodes is a set parameter. The larger the number of episodes, the longer the exploration into the environment.
- *Environment:* contains the set of functionalities behind every step the agent takes. The general template is based on OpenAI Gym for reset, step, and policy. Other functionalities are included to support the policy in assessing the actions. In *reset*, the environment restarts the search and set the initial action and initial state by selecting a candidate pair randomly from a portion size of the SST solution, to avoid frequently selecting the same high scored item. For *step,* the environment updates the current state with the provided action.
- *Policy:* returns the actions to be considered for the next step and the probability of each one, affected by ε. The set of actions returned are the ones compatible with the current state. Each action represents a candidate pair compatible with the current solution represented by a state. In order to measure the compatibility, the LCM scores are calculated for the available features and per relation set. The global score is returned per candidate pair, and the policy selects as the *best*

action the one with the maximum LCM score. Additionally, it selects the best action according to the current state from Q if the state has been experienced, or it will create a new entry.

Algorithm 2 SARSA Main

```
input: env, number episodes episode, $\alpha, \gamma$
output: $Q$
// initialize
1     Set Q(s,a)= initial value-function
2     Set $\alpha$
3     Set $\gamma$
4
5     for each episode
6         reset environment
7         get action a using policy
8         do
9             Take step, set s′,r, terminal state
10            Get a′ from s′ using policy
11            Set $TD_{target} = [r + \gamma Q(s', a')]$
12            Set $TD_{delta} = TD_{Target} - Q(s, a)$
13            Update $Q(s, a) = Q(s, a) + \alpha * TD_{delta}$
14            Update a=a′
15            Update s=s′
16        while terminal state is False
17    return Q
```

Algorithm 3 SARSA Environment: Policy

```
input: observation, QSM, QMM, metric_size, similarity, compatible_pairs
output: LCM scores, terminal state
// initialize
1     Set LCM=0
2     Set $\epsilon$
3     Set terminal state = False
4     Get available pairs
5     if available pairs length is 0
6         Set terminal state = True
7     else
8         Get LCM scores
9         Set nA=number of LCM values
10        Set action probabilities LCM_p =LCM* $\epsilon/nA$
11        Set best_a=max(LCM_p)
12        Update LCM_p[best_a]=LCM_p[best_a]+(1- $\epsilon$)
13    return LCM,  terminal state
```

SARSA searches for solutions for a set of episodes, for which it will run several iterations until the final state is reached. In the main algorithm, new actions and states are found on every step into the environment, and the action-value function Q is updated, to be used in the policy evaluation. The first action is selected randomly from the SST scores to take the agent one step inside the environment. In the environment, with the provided action and state at step t, the policy will calculate the next action probabilities with the LCM scores. As the search continues, and the following episodes are completed, the agent *learns* which combinations of action-states are the best ones according to the rewards received and applied this knowledge to make a better decision in the remaining episodes.

Finally, the algorithm returns the action-value function Q, with the assortment of all states (solutions), actions (candidate pair), and rewards (total score) that modified that solution. The last added solution is the result of an *on-policy* approach, whereas the complete set of solutions represents an *off-policy* procedure.

3.7 EVALUATION

Once the algorithm design is stable, the performance analysis includes a review of the matched features. Consider a decision process to evaluate correctly aligned sketch to metric features labeling a correct or incorrect assignment. There are four possible combinations (categories) organized in a Confusion Matrix, as in Table 6, containing the labels *True Positive, False Positive, False Negative,* and *True Negative.*

		True Condition	
Predicted Condition	**Condition**	**Condition Positive**	**Condition Negative**
	Predicted Positive	True Positive	False Positive
	Predicted Negative	False Negative	True Negative

Table 6 Confusion Matrix

A True statement refers to a correctly classified feature (positive or negative), and the False statement refers to an incorrect classified feature (positive or negative), then a True Negative label out an alignment correctly rejected (Davis & Goadrich, 2006). Based on the Confusion Matrix configuration, four metrics can be derived:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} = \frac{TP}{TP + FP} \tag{3.8}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} = \frac{TP}{TP + FN} \tag{3.9}$$

Then, it is possible to define the concepts of Precision and Recall as:

- *Precision:* the portion of positive features correctly aligned by the algorithm implementation (True Positives)
- *Recall:* the portion of positive features correctly labeled by the algorithm implementation

For the environment performance measure and analysis, as the algorithm implementation is in Python, each algorithm is executed for several steps to stress the environment. The results are stored in a data frame to be exported with the following values: iteration number, execution time in seconds, true positives features, true negatives, false positives, false negatives, and final iteration solution.

CHAPTER 4 RESULTS AND DISCUSSION

The implementation results for the Tabu and SARSA algorithms, as well as their backup functionalities from the Qualitative Analysis and the Score System, are presented in this chapter. Initially, we describe the overall results from the processing and scoring modules. The first subsection (4.1) describes the performance rates in terms of the execution time during the alignment process for each one of the datasets. Secondly, the alignment results are displayed, followed by their discussion. Finally, we highlight some of the limitations encountered.

For *Input Processing*, the sketch maps are processed in the SmartSkeMa framework creating the vectorized features per dataset. The three datasets' attributes are then edited on the Inkscape software. In order to illustrate these steps in Figure 21, the Mailua Ranch processed features are recovered from the SmartSkeMa vectorization to remove the small features later not associated with the smart schema data type.

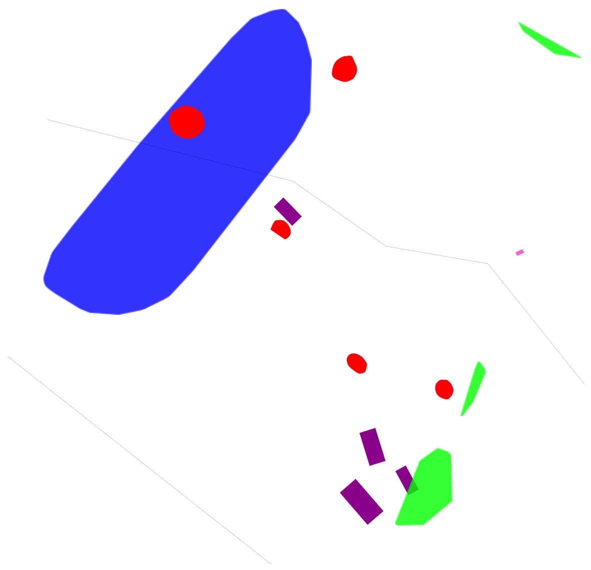


Figure 21 Mailua Ranch vectorized sketch map

For all sketch and metric maps datasets, the SmartSkeMa's *id* and *name* attributes are revised in the XML feature editor to aid the match identification. In Table 7 for the *Marsh* feature in the Mailua ranch dataset, the attribute and values for the sketch and metric map are described:

Table 7 Mailua Ranch Marsh feature attributes

Feature	Attribute	Value
	id	sm_marsh1
	name	sm_marsh1
	smart_skema_type	marsh
Sketch map		
	id	mm_marsh1
	name	mm_marsh1
	smart_skema_type	marsh
Metric map		

In *Qualitative Analysis*, the vectorized sketch maps are handled to calculate the QCN matrix in the SmartSkeMa framework. The qualitative representation process is carried per feature in both maps constructing the relations network and assigning a label from each relation set considered. In Figure 22, the arrangement for some features from the Mailua sketch and metric map data set are displayed:

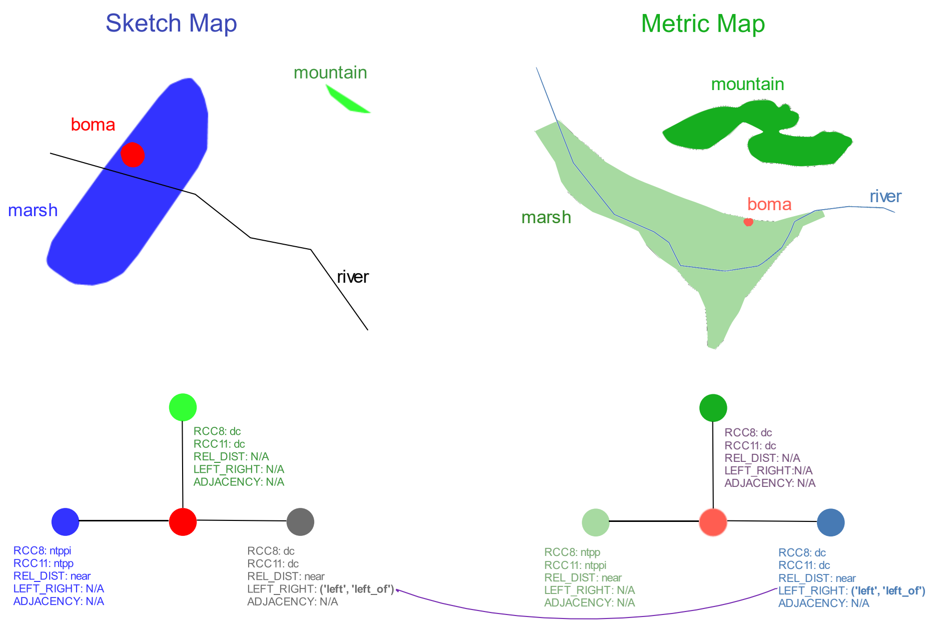


Figure 22 Qualitative representation input maps

Each object is represented by a node with labels describing the spatial relationship between each other according to the spatial calculi detailed in 2.3.1. The purple line connecting the sketch and metric map label, highlights the relation identified for the sketch and metric map in $\mathcal{R}_{left\_right}(boma, river) = 'left, left\_of'$ as this type of similarities give insights about their characteristics in the search space for the matching process. The same mechanism is applied to every feature in the three data sets resulting in three different QCN matrices used by the Scored System and the Qualify Analysis modules to asses and measure the compatibility between candidate pairs. Some calculi return an *N/A* value as the geometry type arguments to calculate the relation are not met. One of the considered calculi, *starVars,* is removed due to incompatibilities with the current inputs during the qualification.

As for the *Scoring System*, the Link Analysis process is executed with the QCN matrices, delivering the score ranking to the SST evaluation function from which we retrieved a subset of candidate pairs highly connected with other features in the search space. In some cases, the output included correctly aligned features; one example is shown in Table 8:

Table 8 Link Analysis sample results

Dataset	Link Analysis Sample
Artificial SVG	'sm815': 'rect815', 'sm817': 'rect817', '
El Remanso	'sm_lake': 'mm_lake', 'sm_marsh1': 'mm_marsh1'
Mailua	'sm_river': 'mm_river', 'sm_road': 'mm_road'

For the LCM scores, we derived two different approaches to evaluate future-promissory candidate pairs to add in the solution: the first heuristic H1 evaluates each pair candidate local compatibility *before adding* during the iteration process returning the scores per each one of the considered relation sets as shown in Table 9. Next, the second heuristic H2 is calculated *after adding* based on the pairs in the current solution, providing a set compatible pairs additionally from the qualitative representation. These differentiations had an essential repercussion in the learning algorithm reviewed in the discussion section.

Table 9 LCM(H1) score sample

Calculi	H1 Score
RCC8	29
RCC11	29
REL_DIST	27
LEFT_RIGHT	21
ADJACENCY	20

Finally, the *Searching algorithms* module is executed. The Tabu Search algorithm implements a scoring system based on LA and the SST. SARSA employs two different configurations: the first one is solely based on the H1 heuristic and SST, the second one analyzes the H2 heuristic and SST. This distinction arises from two separate results returned during the implementation. In the following subsections, the algorithm's results are illustrated in more detail.

4.1 PERFORMANCE

4.1.1 Execution time

Both algorithms execute their tasks for a maximum of 1000 iterations, and a built-in function in Python measures the timing. In general, for a small number of iterations, Tabu is faster, but as the number increases, SARSA shows a recovery using less time despite the number of tasks needed to compute a sub solution.

In the smallest dataset, Artificial SVG, with six features in the sketch map for aligning to 7 features in the metric map, in Figure 23, Tabu takes more time after 350 iterations approximately. SARSA consumes more time in the beginning, but as the search continues, it spends less time computing the results. Due to the backup nature of SARSA, in a lower number of iterations employs more time assessing all the subset solutions (states) to recover the values later when they are recalled.

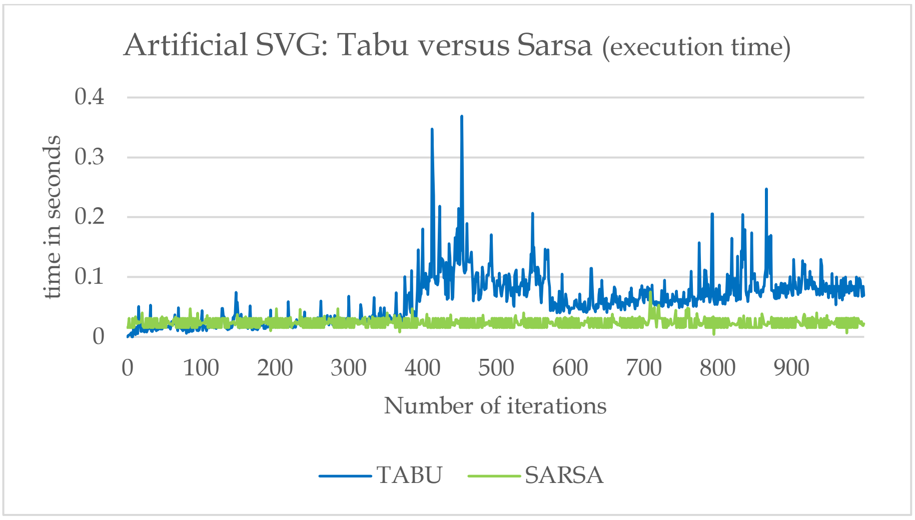


Figure 23 Artificial SVG execution time

For a larger dataset, El Remanso, with 13 features in the sketch map for aligning to 15, in Figure 24, the Tabu algorithm execution time increases with some peaks: as the number of iterations increases, the population of the neighborhood consumes more time as the tabu lists banned the access to compatible candidate pairs. The peaks in SARSA, are related to states in which new items are being explored and added to the solution.

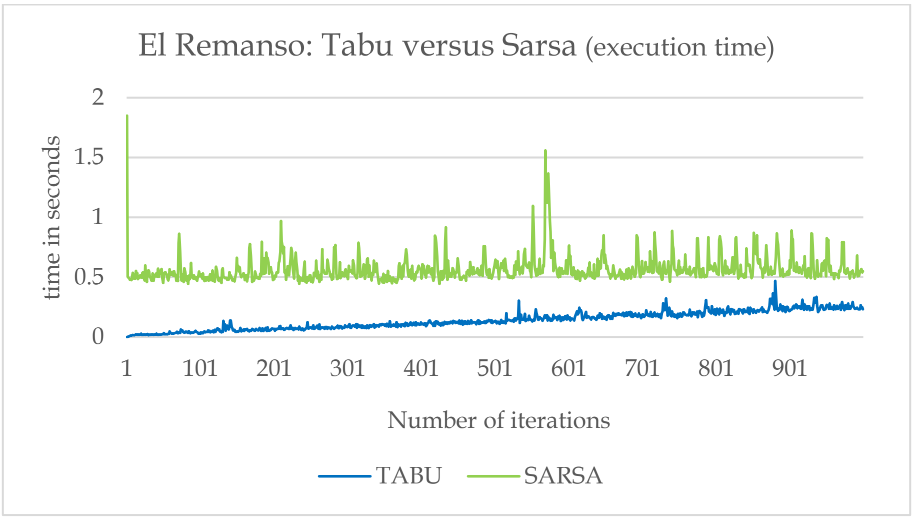


Figure 24 El Remanso execution time

Finally, for the most extensive dataset Mailua Ranch, with 17 features in the sketch map and 106 in the metric map, in Figure 25, the behavior of both algorithms is more visible. In the beginning, SARSA consumes the most considerable amount of time, but it decreases over time. Tabu displays peak points over time due to the available neighborhood update once the tabu lists are full, limiting the access to compatible pairs and encouraging exploration.

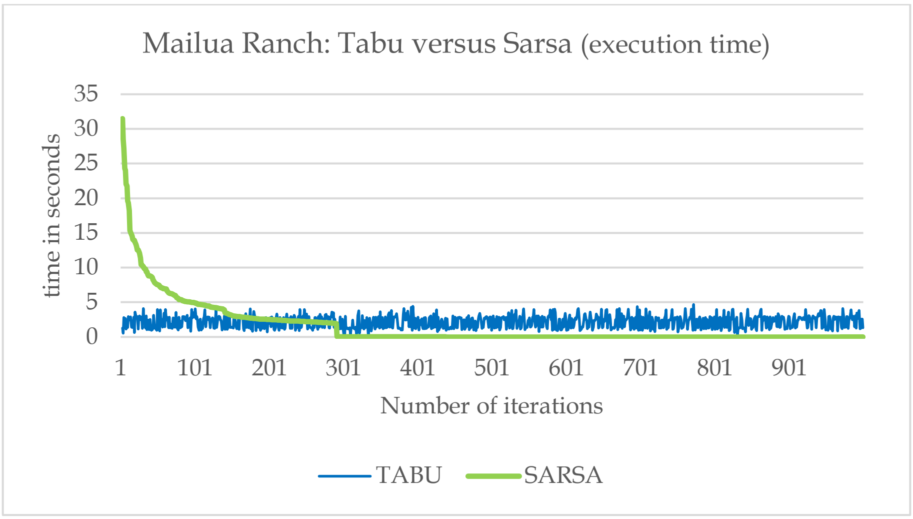


Figure 25 Mailua ranch execution time

4.1.2 Precision and Recall

By using the formulas described in section 3.7, and the results obtained from the algorithms' execution, the precision and recall metrics are calculated. For each one of the datasets, a maximum of 1000 rounds of alignment are executed per algorithm, and the identification of correct alignment is made by code evaluating the number of *True Positives, True Negatives, False Positives,* and *False Negatives* from the output results. In the case of the sketch to metric map alignment, a *True Positive* is every feature correctly aligned with the expected feature from sketch to the metric map, a *False Positive* is every feature wrongly aligned with another feature than the expected one, and a *False Negative* is every feature that should have been aligned, but it is not present in the solution. In the final solution, there are no *True Negatives* to consider, as every feature in the sketch map dataset is obligated to be aligned to at least one feature in the metric dataset.

For the Artificial SVG dataset, the average precision and recall statistics in Table 10 for Tabu are higher for 11.62% and 4.84%. In the alignment results review, the SARSA algorithm varies the precision every number of iterations in between 40% and 100% with a maximum recall of 83%, whereas Tabu from 54% reaches 80% of precision with a maximum recall of 80%, and it does not improve in future iterations once the solution is stable.

Table 10 Artificial SVG: Precision and Recall Results

Algorithm	**Tabu**	**SARSA**
AVG. Precision	80%	68%
AVG. Recall	80%	75%
Min. Precision	60% (0%)*	40%
Max. Precision	80%	100%
Min. Recall	75% (0%)*	67%
Max. Recall	80%	83%

* For the first iteration, the algorithm did not find a solution and returned an empty list

The precision and recall statistics for El Remanso dataset are displayed in two different tables to illustrate the difference between the implementation of SARSA(H1) and SARSA(H2). In Table 11, the average precision in Tabu is 7% higher than SARSA(H2), with average recall differing for 15%, with 99%. In terms of minimum and maximum precision, SARSA(H2) has higher results, returning on every iteration a solution, whereas, for the recall, Tabu aligns 100% of the relevant items selected in contrast to SARSA(H2), with 89%.

Table 11 El Remanso: Precision and Recall Results (H2)

Algorithm	**Tabu**	**SARSA (H2)**
AVG. Precision	58%	51%
AVG. Recall	99%	84%
Min. Precision	55% (0%)*	27%
Max. Precision	58%	73%
Min. Recall	86% (0%)*	75%
Max. Recall	100%	89%

* For the first iteration, the algorithm did not find a solution and returned an empty list

In the course of the approach implementation, we run several times the SARSA algorithm with only the first heuristic scores. The results differ from SARSA(H2) in terms of precision, recall, performance, and spatial configuration. For the SARSA(H1), the average precision displayed in Table 12 is higher than Tabu and SARSA(H2) algorithms with an 84% average recall. With just the first score of the LCMs, we surpassed the 70% window. On the other hand, SARSA(H1) takes more time computing the final solution: for a range of 400 iterations, it takes 35 seconds.

Table 12 El Remanso: Precision and Recall Results (H1)

Algorithm	**Tabu**	**SARSA (H1)**
AVG. Precision	58%	73%
AVG. Recall	99%	84%

The Mailua Ranch data set presents the lowest statistics for both algorithms, as shown in Table 13. For the average precision and recall, the Tabu search is 20% higher, with maximum values reached without variation in future iterations. SARSA(H2) keeps a variation during the search as it explores newer candidate pairs returning in some cases a final solution with *False Negative* results, indicating a requirement for a more substantial number of iterations to explore all candidates' information and return a solution for the non-considered features during the matching process.

Table 13 Mailua Ranch: Precision and recall results

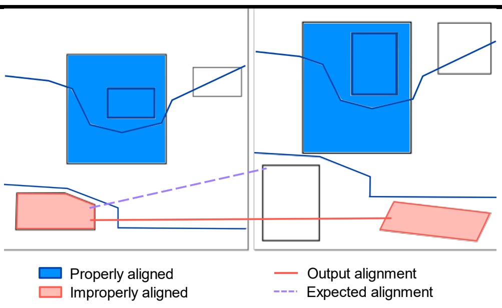
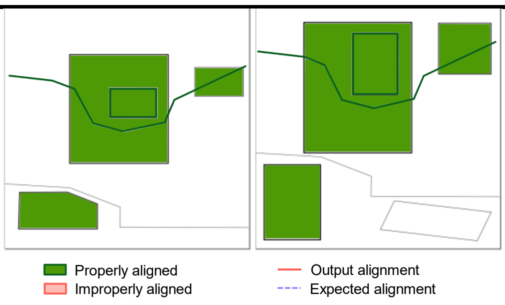
Algorithm	**Tabu**	**SARSA (H2)**
AVG. Precision	31%	11%
AVG. Recall	46%	21%
Min. Precision	0%	0%
Max. Precision	56%	53%
Min. Recall	0%	0%
Max. Recall	100%	89%

4.2 ALIGNMENT RESULTS

The algorithms are executed for a maximum of 1000 iterations, and the output result contains the sketch map feature and an assigned metric map feature. We present three samples from the alignment results for each algorithm, per dataset in which we compare both approaches, followed by the discussion in the next subsection.

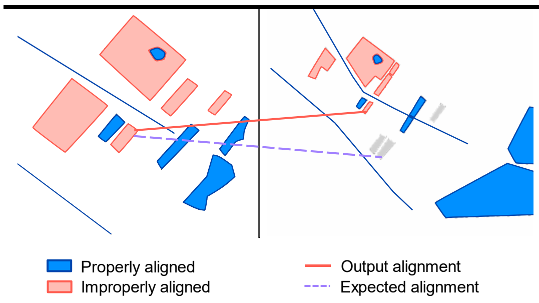
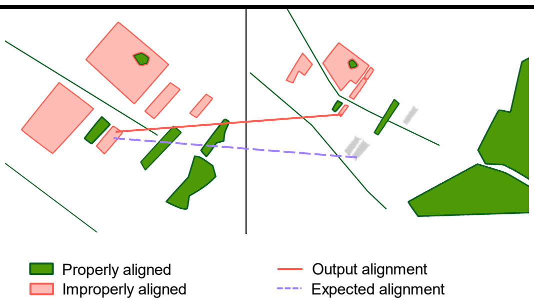
For the smallest dataset displayed in Table 14, SARSA aligned more items in less time for the same number of iterations. Most of the results from Tabu are concentrated in one area with one *False Positive* item returned. The SARSA alignment is more dispersed in the search space aligning 5 of 6 features correctly with one *False Negative.*

Table 14 Artificial SVG: Alignment result sample

Tabu Results		**SARSA Results**	
 Properly aligned Improperly aligned Output alignment Expected alignment		 Properly aligned Improperly aligned Output alignment Expected alignment	
Execution time	0.068991 s	*Execution time*	0.0203 s
Number Iterations	996	*Number Iterations*	996
Number of features aligned	5/6	*Number of features aligned*	5/6
True Positives	4	*True Positives*	5
False Positives	1	*False Positives*	0

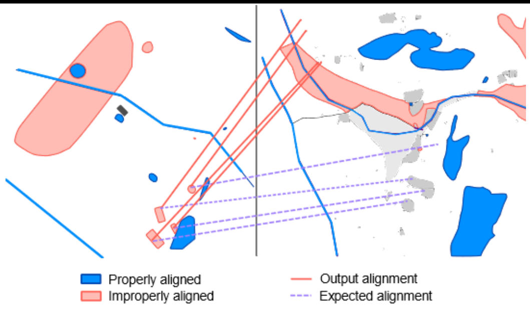
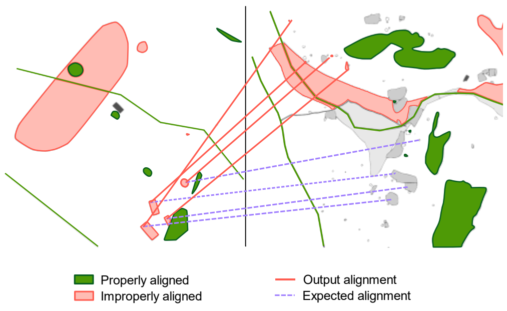
El Remanso dataset doubles the features from the simple sketch map. In this scenario, Tabu is faster and returned one additional *False Positive* aligned feature in half of the time. In the case of SARSA, the additional feature not displayed is a *False Negative*. Both algorithms return a similar solution.

Table 15 El Remanso: alignment result sample

Tabu Results		**SARSA Results**	
 Properly aligned Improperly aligned Output alignment Expected alignment		 Properly aligned Improperly aligned Output alignment Expected alignment	
Execution time	0.25819 s	*Execution time*	0.56543 s
Number Iterations	986	*Number Iterations*	986
Number of features aligned	12/13	*Number of features aligned*	12/13
True Positives	7	*True Positives*	7
False Positives	5	*False Positives*	4

The Mailua Ranch dataset is the largest one with both algorithms returning similar matches for which the differences are noticeable in the not correctly aligned features. Tabu returns an additional *True Positive* feature and SARSA one *False Negative*. The alignment stats are illustrated in Table 16:

Table 16 Mailua Ranch: alignment result sample

Tabu Results		**SARSA Results**	
 Properly aligned Improperly aligned Output alignment Expected alignment		 Properly aligned Improperly aligned Output alignment Expected alignment	
Execution time	3.9044 s	*Execution time*	4.5988 s
Number Iterations	901	*Number Iterations*	901
Number of features aligned	16/16	*Number of features aligned*	15/16
True Positives	9	*True Positives*	8
False Positives	7	*False Positives*	7

4.3 DISCUSSION

The *Score System* module results can be sensitive to the quality of the input sketch map. With the wrong feature type, the LA and SST scores can be corrupted by giving more scores to meaningless objects such as triangles derived from the vectorization process in the SmartSkeMa. We recommend reviewing the quality of the vectorized sketch map and clean the features with a non-compatible object type. Additionally, it is necessary to examine further the dangling factor and the number of iterations for the LA scores, considering factors such as the size of the graph and the geometries type as they influence the size of the initial solution in SST. For the LCM scoring, the outcome for the LCM(H1) provide to the learning algorithm a free exploration inside the environment space as it evaluates the score per pair.

On the other hand, LCM(H2) heuristic is restricted to future promissory pairs in the current solution. The immediate effects of this finding are visible from the alignment results, as the first heuristic recall is higher than the second heuristic. Due to the time constraint, experimenting with the single use of H2 instead of the SST solution for the SARSA algorithm and the corresponding environment configuration update is open for future work.

For the Tabu and SARSA algorithms, the execution time, the number of correct matches, spatial configuration, and complexity are the main aspects evaluated. Tabu works faster with a small number of iterations due to the less elaborated processes required for searching, and with a higher number of iterations, the contribution to the solution decreases to a point in which the output solution is stable, and no significant changes occur. As the number of features to align increases, the longer time will take to arrive at this convergence point. By using two tabu lists instead of one, the exploration was encouraged to add non-high-scored features in the solution, but as the process continues with the same configuration in the search space, nevertheless it is possible to keep receiving the same candidate pairs subset and get into cycling solutions. On the other hand, changing the size of the tabu list can cause the solution to break as the search space is constrained to the number of available features; thus, the minimum size should be related to the length of the initial solution, in our case the SST.

Concerning the alignment results, features aligned by Tabu are attached to a clustering behavior: as displayed in Figure 26, for the Mailua Ranch sketch map, features *A*, *B*, *C* and *R* share their immediate space, but they are not located correctly in the metric map solution (left side). Furthermore, the distribution does not consider the orientation between them: feature *R* is *in front* of the *A*-*B*-*C* neighborhood, and in the output solution, it is *in between* and *far* from the feature *M*.

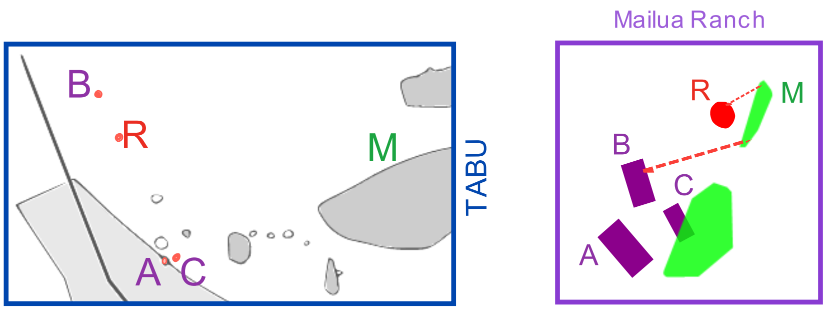


Figure 26 Tabu clustering alignment

Despite SARSA taking a longer time to complete a high volume of tasks initially, the precision of the solution varies over time, coming to values higher than 80% for small to medium-sized datasets for both SARSA(H1) and SARSA(H2). By comparing the results from SARSA(H2) for the same cluster discussed in Tabu for Figure 26, the solution is distributed, not only considering how close the objects are but also is visible the relationships with vicinity features, as illustrated in Figure 27. The *A*-*B*-*C* features are distributed closer to the feature *M* as well as *R*. Moreover, feature *A* should be the one *very far* from *M* in the original arrangement in the Mailua Ranch dataset, with SARSA(H2) returning the displayed spatial configuration correctly.

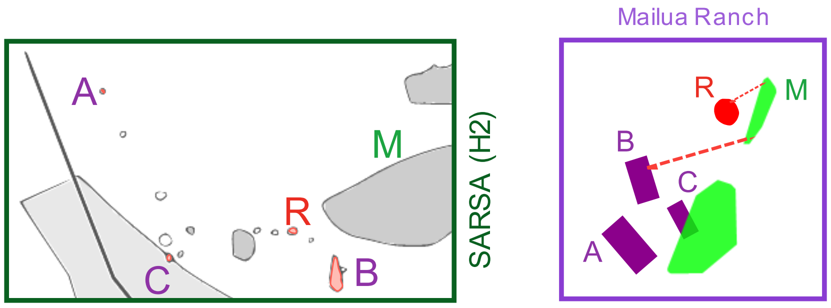


Figure 27 SARSA(H2) clustering alignment

The improvement in H1 from the observation present in H2 has its roots in the state-value function in SARSA, the sub-solution-score backup. As SARSA(H1) explores more the environment, better-rewarded solutions for the same configuration are calculated, and eventually, it selects the best one, contrary to SARSA(H2), that is constrained to a subset of future solutions and may not find an appropriate match in the environment on time. In the results for both configurations in Figure 28, SARSA(H2) ignores aspects of the spatial configuration for feature *C* as it needs to be *the closest* to features *B* and *A*, regardless of the objects in between as SARSA(H1) solution returned.

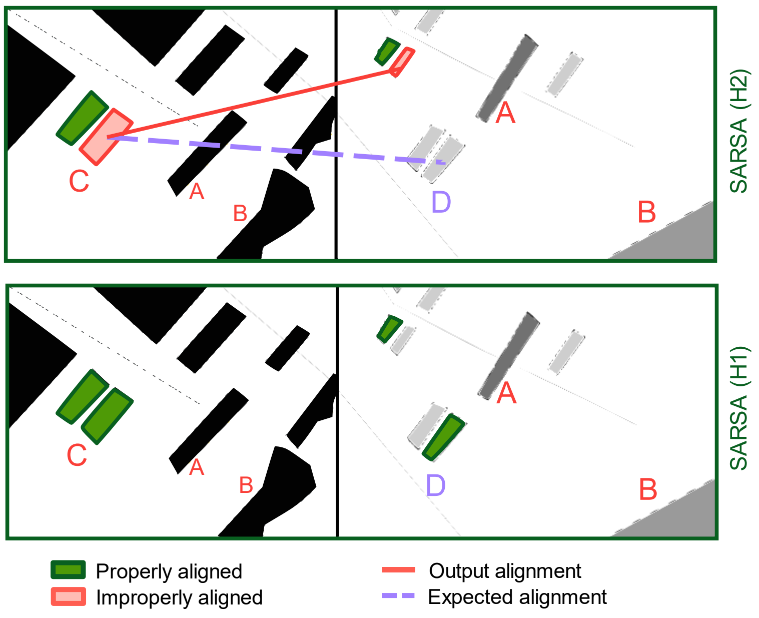


Figure 28 SARSA(H2) compared to SARSA(H1) alignment

For the second heuristic, the spatial configuration is preserved, but the search space is limited to the compatible pairs derived from the similarity matrices evaluation. The algorithm is considering highly connected features as the initial solution, and not all of them exist as promissory candidates according to the H2 criteria. Until both conditions are meet, the algorithm keeps exploring the environment, and thus, the solutions are linked to this restriction. One approach to encourage the search from the second heuristic is to implement the identified subset as an initial solution, instead of looking for highly connected features in SST. Another procedure includes the modification of the Link Analysis process and combining the H2 subset for a hybrid approach.

Correctly aligning features between sketch and the metric maps are linked to the spatial configuration complexity and the number of features in the sketch and the metric map. Both algorithms increased the precision once the vectorized sketch map was cleaned from small polygons as the matching process consists of an exhaustive evaluation of candidate pairs. In order to boost the alignment process, it is crucial to filter the data included in the Qualitative Representation to avoid using resources on meaningless relations.

4.4 LIMITATIONS

One of the limitations encountered for the alignment process is the definition of the number of iterations needed for each algorithm to reach their potential. It is necessary to analyze the conditions to calculate an *equivalent ratio* for the running times as it can be derived from the performance results in El Remanso dataset in which the inflection point for SARSA to overcome the Tabu results is not reached.

Another limitation encountered for the alignment process in the SmartSkeMa framework is the noise caused by vectorized features such small triangles or the split of sketch map features into smaller pieces that are included or excluded in the qualitative segmentation. To overcome this challenge, the use of the module Geometry Editor at the beginning offered a didactic way to digitalize features, but it needed to run a first version of the Vectorize module from which small, not useful features were created. The final procedure was to manually add the objects to artificial vectorized sketch and metric maps by using the Inkscape software and exporting them to SVG, providing the corresponding geoJSON file for the metric map. Additionally, we encountered conflicts in the use of the relation set *starVars* as it faced long execution times and raising errors during the qualitative representation. The temporary solution was to remove *starVars* from the functionality's arguments until a more in-depth analysis is done for understanding the implementation of this calculi during the qualitative analysis. Lastly, because the Sketch to Metric alignment problem is very particular to our interest, the current configurations defined in *OpenAI Gym* to run learning algorithms had limited use, leading to the implementation of a new setting based on the predefined templates.

CHAPTER 5 CONCLUSIONS

For the Sketch to Map alignment problem, we proposed the use of two different algorithms and a scoring system to evaluate each possible candidate match. The workflow combines five main modules: input processing, qualitative analysis, score system, search algorithms, and evaluation. In the input processing module, we provided the SmartSkeMa framework with the sketch and metric maps to be processed and vectorized, for which we manually edit the SmartSkeMa attributes, providing three input maps with different levels of complexity in terms of the number of features and spatial configurations. In the qualitative analysis component, we outline the process for evaluating compatibility between each candidate pair in terms of consistency of the constrained network and feature type with the retrieved QCN using these functionalities during the scoring process. The Score System offers the possibility to calculate four scores that can be used combined or some separately: the Link Analysis (LA) score provided information about the level of connectivity of each feature in our search space, Spectral Solution Technique (SST) processes the LA ranking to return a set of highly compatible features giving us an initial solution, and finally, the two Heuristics Scores based on the Local Compatibility Matrices deliver a measure for forthcoming solutions derived from each candidate pair. Then, the different scores are used in the Searching Algorithms module which consists of two implementations: a new Tabu Search incorporating LA and SST scores, returning a set of features from the sketch and metric maps from iteratively evaluating the compatibility of each pair candidate and banning time to time the ones considered to be out of the solution or recently added; on the other hand, the SARSA algorithm by using SST and LCM scores experiences several sub solutions with different sizes, and selects over time the ones with the highest scores to construct a final solution based on the best possible combination of subset matching solutions.

At the end of this thesis, we analyzed the results of the workflow and mentioned the limitations encountered. The Qualitative Analysis module helped to accurately identify the compatibility between features, visible in the results as the output solutions are coherent regarding the type and the shared constraints. Secondly, the Score System delivered on each call the evaluation measurements making use of the SST initial solution showing the influence of highly connected pairs in the search space, as well as the calculus of Local Compatibility Matrices with the evaluation of the corresponding heuristics per candidate set. The new Tabu algorithm surpasses the statistics of average precision for the SARSA algorithm with LCM(H2) (80% vs. 68% smallest dataset), increasing the gap as the number of features increases (31% vs. 11% largest dataset). Nevertheless, analyzing the solutions derived from both algorithms, it is vital to notice that the False Positives features for SARSA are *closer* to the original spatial configuration in the sketch map, especially for the implementation with only LCM(H1), and the maximum precision of the algorithm varies as the number of iterations changes, reaching 100% in specific cases indicating a relevant percentage of *True Positives* matches compared to Tabu.

The Tabu solution is faster in a shorter number of iterations, more straightforward and offers higher results in terms of precision, but on the other hand, the SARSA performance improves over time with consistent spatial distribution compared to Tabu. As the number of iterations goes on, the dynamic programming algorithm can offer a range of matches, giving highlights about how the search is being approached thanks to the backup of the subset solutions and their scores, whereas for Tabu once the solution is stable, it will be returned repeatedly over the time without further exploration of the search space or improvement. Two main configurations for the policy calculus in the learning algorithm were implemented: solely the first heuristic *and* with both LCM(H1) and LCM(H2) heuristics, returning higher or lower precision and recall statistics than Tabu, which leaves the door open to implement different LCM scores configurations in the same environment.

In conclusion, the main contributions of this master thesis are the performance improvement for QCNs in large scale datasets, and the support during the matching process with a global overview of the spatial configuration described on them by including the implementation of four different scores: link analysis, spectral solution, and two heuristics from the Local Compatibility Matrix. In SARSA, the Q values summarize the information about the SST and LCM scores allowing the search to invest the time saved, exploring more the search space updating information about the candidate pairs and sub solutions, whereas Tabu only uses information from the immediate neighborhood. Next, we implemented the module for the construction of the LCM scoring: the search results retrieved from the implementation of both heuristic scores returns consistent information about the local spatial configuration for the pairs belonging to the subset solution in the SARSA algorithm avoiding wrong alignments with a less complete solution compared to Tabu, which returns a more complete set of matched features by allowing mistakes.

Finally, we implemented two different searching algorithms with distinct advantages: Tabu is more straightforward and works in this case for immediate analysis of alignment results. For long-term, more spatially structured matches, SARSA by taking advantage of the backup of subset solutions and learning from the exploration process in the search space, presents a selection of pairs with coherent arrangements with reference to other features.

5.1 FUTURE WORK

During the workflow implementation, different ideas to improve the current solution arose from the use of the score modules to the learning algorithms. One approach is to analyze the features clusters retrieved from the Spectral Solution Technique and review how the different clusters can be labeled to work as subset solutions to limit the search space and apply the Local Compatibility Matrices score to answer the question: *Does clustering identification or limiting the search to identified clusters improve the alignment solution?*

Secondly, the inclusion of the qualitative calculus *starVars* into the algorithm should be reviewed as *orientation type* relationships can add value to the matching score during the search, enlarging the subset of compatible pairs to keep improving the spatial configuration. Lastly, new dynamic programming algorithm implementations compatible with the characteristics of the QCN should be considered given the potential found in reinforcement learning algorithms in the graph matching problem.

CHAPTER 6 REFERENCES

Anaconda. (2020). Anaconda Distribution. Retrieved January 2, 2020, from Anaconda Documentation website: https://docs.anaconda.com/

Battiti, R., & Protasi, M. (2001). Reactive local search for the maximum clique problem. *Algorithmica (New York)*. https://doi.org/10.1007/s004530010074

Battiti, Roberto, & Tecchiolli, G. (1994). The Reactive Tabu Search. *ORSA Journal on Computing*, *6*(2), 126–140. https://doi.org/10.1287/ijoc.6.2.126

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym*. Retrieved from https://arxiv.org/pdf/1606.01540.pdf

Broelemann, K. (2011). A System for Automatic Localization and Recognition of Sketch Map Objects. In: Wang, J., Broelemann, K., Chipofya, M., Schwering, A., and Wallgrün, J.-O. (eds.). *COSIT 2011 Workshop on Understanding and Processing Sketch Maps*, 11–20. Belfast, Maine. Heidelberg, AKA GmbH.

Broelemann, Klaus, & Jiang, X. (2013). A region-based method for sketch map segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7423 LNCS*, 1–14. https://doi.org/10.1007/978-3-642-36824-0_1

Broelemann, Klaus, Jiang, X., & Schwering, A. (2016). Automatic understanding of sketch maps using context-aware classification. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2015.09.037

Bunke, H. (2000). Recent developments in graph matching. *Proceedings - International Conference on Pattern Recognition*. https://doi.org/10.1109/ICPR.2000.906030

Bunke, H., & Jiang, X. (2000). *Graph Matching and Similarity*. https://doi.org/10.1007/978-1-4615-4401-2_10

Ceri, S., Bozzon, A., Brambilla, M., Della Valle, E., Fraternali, P., Quarteroni, S., … Quarteroni, S. (2013). An Introduction to Information Retrieval. In *Web Information Retrieval*. https://doi.org/10.1007/978-3-642-39314-3_1

Chipofya, M. C., Schultz, C., & Schwering, A. (2016). A metaheuristic approach for efficient and effective sketch-to-metric map alignment. *International Journal of Geographical Information Science*. https://doi.org/10.1080/13658816.2015.1090000

Chipofya, M, Jan, S., Schultz, C., & Schwering, A. (2017). Towards Smart Sketch Maps for Community-driven Land Tenure Recording Activities. *Agile*. Retrieved from https://agile-online.org/conference_paper/cds/agile_2017/shortpapers/155_ShortPaper_in_PDF.pdf

Chipofya, Malumbo. (2018). *Matching Qualitative Constraint Networks with Online Reinforcement Learning*. https://doi.org/10.29007/1g5q

Chipofya, Malumbo, Schwering, A., & Binor, T. (2013). Matching qualitative spatial scene descriptions á la Tabu. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-45111-9_34

Chipofya, Malumbo, Wang, J., & Schwering, A. (2011). Towards cognitively plausible spatial representations for sketch map alignment. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-23196-4_2

Cho, M., Lee, J., & Lee, K. M. (2010). Reweighted random walks for graph matching. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-15555-0_36

Conte, D., Foggia, P., Sansone, C., & Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*. https://doi.org/10.1142/S0218001404003228

Cook, D. J., & Holder, L. B. (2006). Mining Graph Data. In *Mining Graph Data*. https://doi.org/10.1002/9780470073049

Cour, T., Srinivasan, P., & Shi, J. (2007). Balanced graph matching. *Advances in Neural Information Processing Systems*. https://doi.org/10.7551/mitpress/7503.003.0044

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/1143844.1143874

Dehmer, M. (2008). Information processing in complex networks: Graph entropy and information functionals. *Applied Mathematics and Computation*. https://doi.org/10.1016/j.amc.2007.12.010

Dehmer, M., Emmert-Streib, F., & Kilian, J. (2006). A similarity measure for graphs with low computational complexity. *Applied Mathematics and Computation*. https://doi.org/10.1016/j.amc.2006.04.006

Emmert-Streib, F., Dehmer, M., & Shi, Y. (2016). Fifty years of graph matching, network alignment and network comparison. *Information Sciences*. https://doi.org/10.1016/j.ins.2016.01.074

Foggia, P., Percannella, G., & Vento, M. (2014). Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*. https://doi.org/10.1142/S0218001414500013

Freska, C. (1991). Qualitative Spatial Reasoning. *Cognitive and Linguistic Aspects of Geographic Space*, 361–372.

Gendreau, M., & Potvin, J.-Y. (2005). Tabu Search. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (pp. 165–186). Montreal: Springer.

Glover, F. (1986). Future paths for Integer Programming. *Computers and Operations Research*, *13*(5), 533–549. https://doi.org/http://dx.doi.org/10.1016/0305-0548(86)90048-1

Glover, F. (1989a). Tabu Search - Part I. *Orsa Journal on Computing*, *1*(3), 190–206.

Glover, F. (1989b). Tabu Search - Part II. *Journal on Computing*, *1* and *2*(December 2018), 190–206, 4-32,. https://doi.org/10.1287/ijoc.2.1.4

Glover, F., Taillard, E., & Taillard, E. (1993). A user's guide to tabu search. *Annals of Operations Research*, *41*(1), 1–28. https://doi.org/10.1007/BF02078647

Jan, S., Chipofya, M., Murcia, C., Schwering, A., Schultz, C., Karamesouti, M., … Wayumba, R. (2018). *its4land Derivable 3.3: technical report*. Retrieved from https://its4land.com/wp-content/uploads/2018/08/its4land_deliverable_D3.3.pdf

Jan, S., Schwering, A., Schultz, C., & Chipofya, M. (2015). RCC11: A Finer Topological Representation for the Alignment of Regions in Sketch Maps. *28th International Workshop on Qualitative Reasoning (QR-2015).*

Karamesouti, M., Schultz, C., Chipofya, M., Jan, S., Murcia Galeano, C. E., Schwering, A., & Timm, C. (2018). The Maasai of Southern Kenya domain model of land use. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. https://doi.org/10.5194/isprs-annals-IV-4-105-2018

Lee, J. H., Renz, J., & Wolter, D. (2013). StarVars-effective reasoning about relative directions. *IJCAI International Joint Conference on Artificial Intelligence*.

Leordeanu, M., & Hebert, M. (2005). A spectral technique for correspondence problems using pairwise constraints. *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2005.20

Ligozat, G. (2005). Categorical methods in qualitative reasoning: The case for weak representations. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/11556114_17

Ligozat, G. (2013). Qualitative Spatial and Temporal Reasoning. In *Qualitative Spatial and Temporal Reasoning*. https://doi.org/10.1002/9781118601457

Murcia, C. (2018). *Automatic understanding of sketch maps using deep learning and computer vision*. University of Muenster.

Ng, R. T., & Han, J. (2002). CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*. https://doi.org/10.1109/TKDE.2002.1033770

NumPyCommunity. (2020). NumPy. Retrieved January 3, 2020, from https://numpy.org/

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998a). The PageRank Citation Ranking: Bringing Order to the Web. *MIT Press*. https://doi.org/10.1109/IISWC.2012.6402911

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998b). The PageRank Citation Ranking: Bringing Order to the Web. *World Wide Web Internet And Web Information Systems*. https://doi.org/10.1.1.31.1768

Pavony, G. R. M. (2000). *Los años del cambio: historia urbana de Bogotá, 1820-1910*. Bogotá, Colombia: Pontificia Universidad Javeriana.

QGISORG. (2002). QGIS - The Leading Open Source Desktop GIS. Retrieved January 3, 2020, from https://www.qgis.org/es/site/about/index.html

Randell, D. A., Cui, Z., & Cohn, A. G. (1992). A Spatial Logic based on Regions and Connection. *3rd International Conference On Knowledge Representation And Reasoning*. https://doi.org/10.1.1.35.7809

Ratitch, B., & Precup, D. (2015). Sparse Distributed Memories for On-Line Value-Based Reinforcement Learning Bohdana. *Lecture Notes in Computer Science*, (July). https://doi.org/10.1007/978-3-540-30115-8

Robson, E., van Kerkhoff, L., & Cork, S. (2019). Understanding citizen perceptions of the Eastern Hills of Bogota: a participatory place-based ecosystem service assessment. *Urban Ecosystems*. https://doi.org/10.1007/s11252-018-0739-9

Saad, E. (2011). Bridging the gap between reinforcement learning and knowledge representation: A logical off- and on-policy framework. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6717 LNAI*, 472–484. https://doi.org/10.1007/978-3-642-22152-1_40

Schwering, A., & Wang, J. (2010). SketchMapia–A framework for qualitative mapping of sketch maps and metric maps. *Las Navas 20th Anniversary Meeting on Cognitive and Linguistic Aspects of Geographic Spaces*.

Schwering, A., Wang, J., Chipofya, M., Jan, S., Li, R., & Broelemann, K. (2014). SketchMapia: Qualitative Representations for the Alignment of Sketch and Metric Maps. *Spatial Cognition and Computation*. https://doi.org/10.1080/13875868.2014.917378

Scivos, A., & Nebel, B. (2005). The finest of its class: The natural point-based ternary calculus $\mathscr{L}\mathfrak{R}$, for qualitative spatial reasoning. *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*. https://doi.org/10.1007/978-3-540-32255-9_17

Štěpánková, O. (1992). *An introduction to qualitative reasoning*. https://doi.org/10.1007/3-540-55681-8_47

Sutton, R., & Barto, A. (1999). Reinforcement Learning: An Introduction. *Trends in Cognitive Sciences*. https://doi.org/10.1016/s1364-6613(99)01331-5

Sutton, R., & Barto, A. (2018). Reinforcement Learning— An Introduction. In *Proceedings of the Annual Conference of the Western College Reading Association* (Second). https://doi.org/10.1080/24699365.1977.11669658

The Python Software Foundation. (2003a). 8.3. collections — High-performance container datatypes. Retrieved January 3, 2020, from https://docs.python.org/2/library/collections.html

The Python Software Foundation. (2003b). 9.7. itertools — Functions creating iterators for efficient looping. Retrieved January 3, 2020, from https://docs.python.org/2/library/itertools.html

The Python Software Foundation. (2009). What is Python? Retrieved January 3, 2020, from Python.org website: https://docs.python.org/3/faq/general.html#what-is-python

Wallgrün, J. O., Wolter, D., & Richter, K. F. (2010). Qualitative matching of spatial information. *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*. https://doi.org/10.1145/1869790.1869833