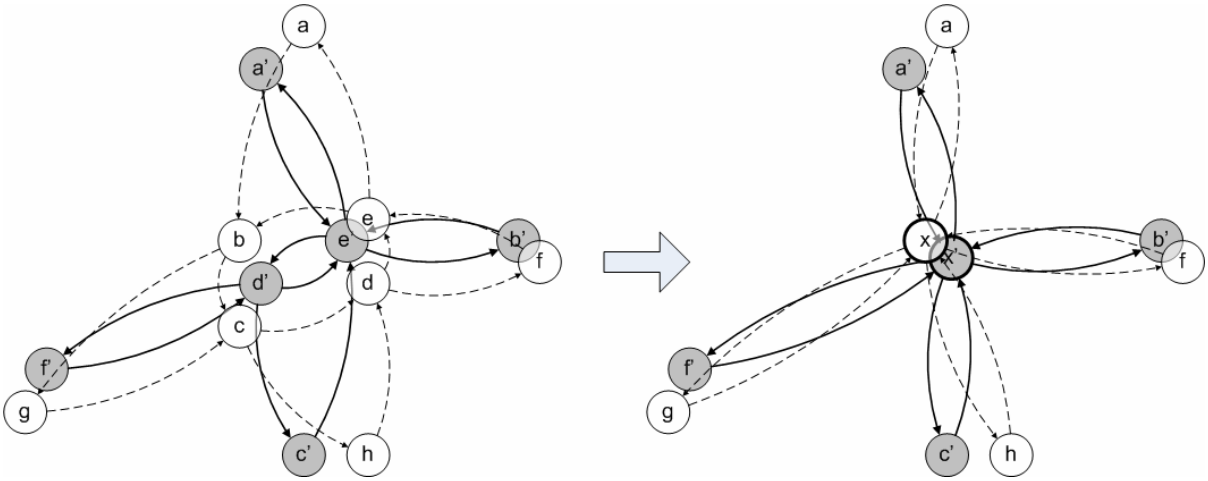


Matching Geo-Coded Graphs

Michael Balmer, IVT, ETH Zurich
Michael Bernard, IVT, ETH Zurich
Kay W. Axhausen, IVT, ETH Zurich

Conference paper STRC 2005

STRC | 5th Swiss Transport Research Conference
Monte Verità / Ascona, March 9-11, 2005

Matching Geo-Coded Graphs

Michael Balmer
IVT
Zurich

Phone: +41-1-633 27 80
Fax: +41-1-633 10 57
email:
balmer@ivt.baug.ethz.ch

Michael Bernard
IVT
Zurich

Phone: +41-1-633 39 43
Fax: +41-1-633 66 94
email:
bernard@ivt.baug.ethz.ch

Kay W. Axhausen
IVT
ETH Zurich

Phone: +41-1-633 39 43
Fax: +41-1-633 10 57
email:
axhausen@ivt.baug.ethz.ch

March 2005

Abstract

In transportation planning and modelling feasible transportation networks are crucial. To be useful, the networks have to fulfil certain requirements: first, the geographical locations of network elements (typically nodes and links) have to be accurate; second, the given attributes (i.e. number of lanes, length, allowed speed, and so on) of the network elements should hold correct information; and—particularly for traffic path finding algorithms—any given network should be constructed such, that every node is accessible by any other node via at least one path.

Unfortunately, in practice there is no guarantee that these three requirements are fulfilled. At the same time often many different networks are available for the same geographical region. These networks often can differ in their emphasis, resulting in differences such as the resolution of the network, the correctness of the geographical locations and the correctness / completeness of the given attributes.

To deal with this problem, one is required to match different networks of the same region so that attributes can be easily shared between the given networks.

In this paper some approaches for network matching are described and compared. Unlike other approaches attributes of the nodes and links are not used as part of the matching algorithm, since they are unreliable. The problem is thus reduced to two directed graph with the addition of spatial information–a geo-coded digraph.

Keywords

network matching – graph algorithm – transportation planning – Swiss Transport Research Conference – STRC 2005 – Monte Verità

1. Introduction

The question about matching different networks describing the same region appears in varies topics, like cognitive recognition science (i.e. Fitch *et al.*, 2002), computer vision (i.e. Christmas *et al.*, 1995), etc. Even so, the benefit of matching different networks varies, i.e. in cognitive recognition science matching is used to find similarities and to extract patterns for learning algorithms (i.e. Christmas, 1995) or detection of changes of sewer tunnels in history (i.e. Pendyala, 2002).

In transportation planning and modelling feasible transportation networks are crucial. To be useful, the networks have to fulfil certain requirements: first, the geographical locations of network elements (typically nodes and links) have to be accurate; second, the given attributes (i.e. number of lanes, length, allowed speed, and so on) of the network elements should hold correct information; and—particularly for traffic path finding algorithms—any given network should be constructed such, that every node is accessible by any other node via at least one path. Unfortunately, in practice there is no guarantee that these three requirements are fulfilled. At the same time many different networks are available for the same geographical region. These networks often can differ in their emphasis, resulting in differences such as the resolution of the network, the correctness of the geographical locations and the correctness / completeness of the given attributes. To deal with this problem, one is required to match different networks of the same region so that attributes can be easily shared between the given networks. To share attributes an appropriate matching of the given networks is needed.

To respect the different benefits of matching networks, this paper shows first approaches how any kind of geo-coded networks can be defined as a geo-coded digraph (a directed graph based on a coordinate system) and it also shows how a matching can be done while the only input are geo-coded digraphs. Unlike other approaches (i.e. Waldner, 2005) attributes of the nodes and links are not used as part of the matching algorithm, since they are unreliable. The problem is thus reduced to two directed graph with the addition of spatial information–a geo-coded digraph.

To describe a geo-coded digraph some definitions have to be made shown in Section 2. The following section deals with pre-conditions and classifications of a graph and its elements. Also some preparations are made such that the resulting geo-coded digraph can be used by the matching algorithm. It is described in detail in Section 4. Then the algorithm is tested against sets of test graphs shown in Section 5 to measure the degree of success. The paper finishes with a summary and outlook.

2. Definitions

2.1 Geo-Coded Digraphs (GCDG)

A geo-coded graph $GCG(V,E)$ consists of a set of vertices V and a set of edges E. Each node is written by a lower case letter $i, j, \ldots \in V$ and each edge by a pair of nodes $(i,j) \in E$. Additionally, each vertex i of a GCG holds a coordinate pair (x_i, y_i) based on a given coordinate system.

In a geo-coded digraph $GCDG(V,E)$ each edge (i,j) of the graph has a defined direction, where i is the source vertex of the edge, j the sink, resp. Each $GCG(V,E')$ can be mapped to a $GCDG(V,E)$ where each undirected edge can be replaced by two edges in opposite direction:

$$(i,j) \in E' \mapsto \{(i,j),(j,i)\} \in E, \quad \forall i,j \in V$$

Since all nodes in a $GCDG(V,E)$ have a defined coordinate, the angle $\varphi_{(i,j)}$ between the abscissa and an edge $(i,j) \in E$ is defined by

$$\cos(\varphi_{(i,j)}) = \frac{x_j - x_i}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}, \quad \varphi_{(i,j)} =]-\pi, \pi].$$

The cosine is a periodic function. To get a unique angle, we just define the range inside one period. Note, that if the source and sink of the edge have the same coordinates, this formula is not valid (division by zero). Therefore we need to add the following constraint:

$$\begin{aligned} &\forall (i,j) \in E: \quad x_i \neq x_j \vee y_i \neq y_j \\ &\Rightarrow \forall i \in V: \quad (i,i) \notin E \end{aligned}$$

This constraint is only related to the edges of the graph. It is still allowed to have vertices with the same coordinates as long as they are not connected by an edge. It includes also that no edge is allowed with the same vertex as the source and the sink.

With the knowledge of the angle of each edge in a $GCDG$ the set of incident edges $\{(i,j)\} \cup \{(k,i)\}$ of a vertex i can be ordered by the following rule:

$$\varphi_{i,j} < \varphi_{i,k} \rightarrow \begin{cases} (i,j) < (i,k) \\ (i,j) < (k,i) \\ (j,i) < (i,k) \\ (j,i) < (k,i) \end{cases}; \varphi_{i,j} > \varphi_{i,k} \rightarrow \begin{cases} (i,j) > (i,k) \\ (i,j) > (k,i) \\ (j,i) > (i,k) \\ (j,i) > (k,i) \end{cases}; \varphi_{i,j} = \varphi_{i,k} \rightarrow \begin{cases} (i,j) = (i,k) \\ (i,j) = (k,i) \\ (j,i) = (i,k) \\ (j,i) = (k,i) \end{cases}$$

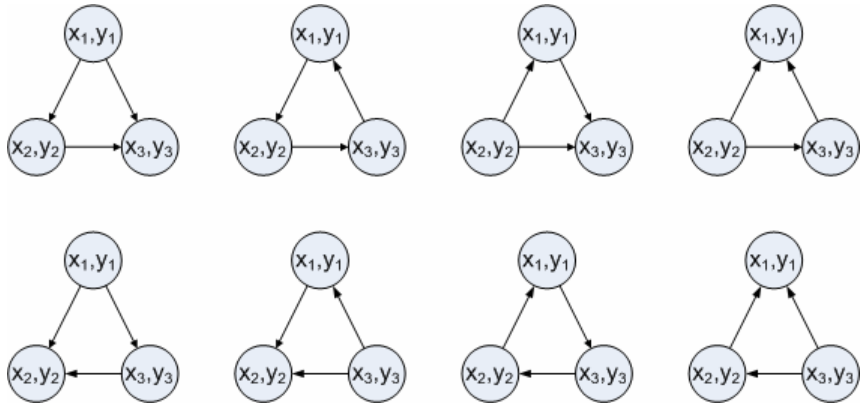
Note that for a given edge (j,i) the angle $\varphi_{(i,j)}$ is defined as the angle of edge (i,j). Since the order is based on an angle between $]-\pi,\pi]$, we can extend it to a ring structure: The last edge in the ordered list defined above therefore can be seen as $<$ (can be interpreted as "is right to") the first edge.

2.2 Isomorphism, Topology and Geo-Topological Similarity

In graph theory isomorphism is well defined (West, 2000). Formally, two graphs $G(V,E)$ and $G(V',E')$ with graph vertices $V = V' = \{1,2,\ldots,n\}$ are said to be isomorphic if there is a permutation p of V such that $\{(i,j)\}$ is in the set of graph edges E if and only if $\{(p(i),p(j))\}$ is in the set of graph edges E'. Determining if two graphs are isomorphic is thought to be neither an NP-complete problem nor a P-problem, although this has not been proved (Skiena, 1990). In fact, there is a famous complexity class called "graph isomorphism complete" which is thought to be entirely disjoint from both NP-complete and from P. However, a polynomial time algorithm is known when the maximum vertex degree (the number of edges incident to a vertex) is bounded by a constant (Luks, 1982). The definition of isomorphism shows that vertices are not bounded to a position. In geo-coded digraphs permutations cannot be done since their vertices are geographically bounded.

Also the concept of topology is known in graph theory. An unlabeled transitive digraph (Sloane, N. J. A. and S. Plouffe, 1995) with n vertices is called topology. Again, vertices are not geographically bounded in this definition. Therefore we need to introduce the concept of "geo-topology". For example, if we define a digraph with three vertices and three directed edges there is exactly one topology. Considering a $GCDG(V,E)$, then there are $2^{|E|} = 2^3 = 8$ different topologies shown in Figure 1.

Figure 1 Eight geo-topologically different geo-coded digraphs



It is easy to find out if two geo-coded digraphs are geo-topologically equal: Two geo-coded digraphs $GCDG(V,E)$ and $GCDG(V',E')$ are said to be "geo-topologically equal" if there is a permutation p of V such that $\{(i,j)\}$ is in the set of graph edges E only if $\{(p(i),p(j))\}$ is in the set of graph edges E' and $\forall i \in V : x_i = x_{p(i)} \wedge y_i = y_{p(i)}$. With the additional information about coordinates of the vertices, the complexity is $O(|V|+|E|)$.

We also can easily add fuzziness into this definition: Two geo-coded digraphs $GCDG(V,E)$ and $GCDG(V',E')$ are said to be "geo-topologically r-similar" if there is a permutation p of V such that $\{(i,j)\}$ is in the set of graph edges E if and only if $\{(p(i),p(j))\}$ is in the set of graph edges E' and $\forall i \in V : r^2 \geq (x_{p(i)} - x_i)^2 + (y_{p(i)} - y_i)^2$. In other words, geo-topological r-similarity between two geo-coded digraphs are given, when a mapping of the vertices exists such that the distance between each mapping pair is not bigger than radius r. If $r \to \infty$, the question about geo-topological r-similarity is then equal to the question about isomorphism.

2.3 Street Network

In transport planning street networks are modelled by graphs. Typically intersections are defined by one or more vertices (also called nodes) holding coordinates, connected by unidirectional and/or bidirectional edges (also called links or lines). Additionally also polylines can be defined to connect the given nodes. Polylines normally have the purpose to model curves more precisely according to the real street. By definition, the "nodes" inside a polyline (called polypoints) are not a model for an intersection. Therefore, we could say that available street networks model each given intersection/bifurcation by nodes, while

polypoints only defines curves of a street connecting two intersections. In reality that is not always true: sometimes nodes of the network do not necessarily define an intersection.

However, we do not need to distinguish between intersection nodes and other nodes. We just define street network nodes as a geo-coded vertices and network links/polylines as directed edges. If a link/polyline is defined as bidirectional we define two directed edges in opposite direction. Note, that we do not consider the additional points in a polyline.

Note that many additional attributes for street network elements can be defined. For example a node can include turning rules, a type, etc. Links and polylines typically hold number of lanes, length, capacities, and so on. We do not use any of that information in the following sections but we keep the mapping between links and edges, nodes and vertices resp. But we need to consider one fact about given networks: Because of different requirements sometimes additional links and nodes are included in networks that do not exist in reality. Typical examples are connector links (see VISUM, http://www.ptv.de or De Palma *et al.*, 1997) that are used to connect zones/municipalities to the networks. It is important to remove those network elements before mapping the network to a graph.

With the above described definitions we are now able to create $GCDG(V, E)$ out of any kind of street networks. Even more, with an appropriate mapping description, we are able to create geo-coded digraphs out of other networks like railroads, rivers, oil-pipelines, etc.

3. Graph Pre-Conditions, Vertex Classification and Reduction Rules

3.1 Graph Pre-Conditions

We now want to match two given geo-coded digraphs $GCDG(V,E)$ and $GCDG(V',E')$ like described in Section 2. First of all, matching does only make sense if the two graphs describe the same region and the same network. It is also necessary that both geo-coded digraphs refer to the same coordinate system. Even more, we need to assure that both graphs hold common parts of information. For example, no match will be possible if one graph models only the highways whereas the other one only describes side roads. Let us summarize those pre-conditions.

Two geo-coded digraphs $GCDG(V,E)$ and $GCDG(V',E')$ that are to be matched have to fulfil the following pre-conditions:

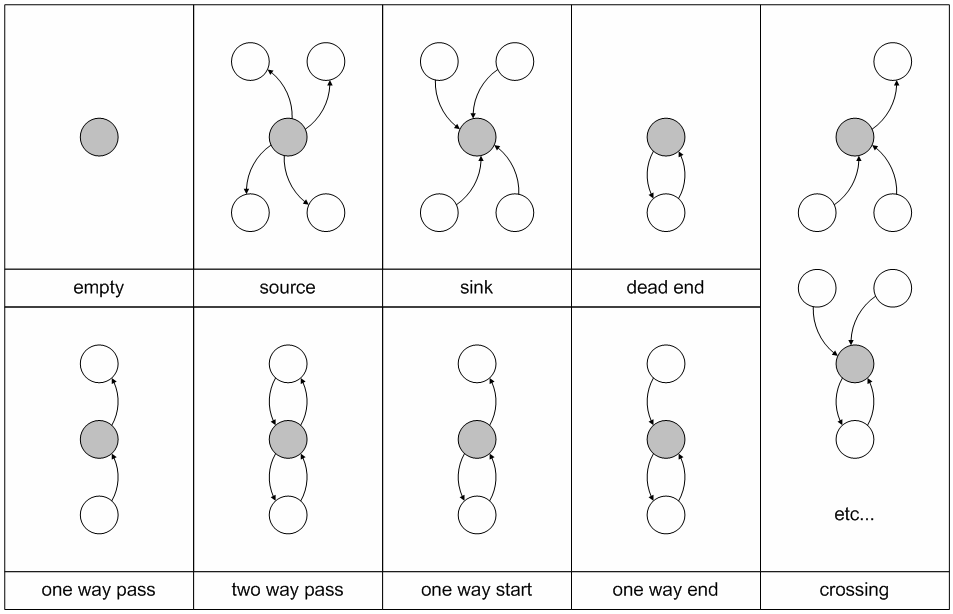
1. $\forall i \in V \wedge \forall j \in V' : x_i, y_i, x_j, y_j$ refer to the same coordinate system.
2. They describe the same type of network system.
3. They describe the same region.
4. $\exists GCDG(V_s,E_s) \subseteq GCDG(V,E) \wedge \exists GCDG(V_s',E_s') \subseteq GCDG(V',E')$, such that $GCDG(V_s,E_s)$ and $GCDG(V_s',E_s')$ hold the same information of the network.

3.2 Vertex Classification and Reduction Rules

In principle two geo-coded digraphs can be matched if they are geo-topologically similar as defined in Section 2.2. This includes that both graphs have to have the same number of edges and vertices. But in fact *GCDG*'s created by given street networks usually differs in the amount of edges and vertices, even if they fulfil the above given pre-conditions. Therefore we need to "reduce" the graphs in a way by which they do not loose matching information. In street networks we need to assure that we do not loose information about the intersections and the streets connecting those intersections since those elements are holding all topological information of the graph. In a first step, we can partially reduce a graph without concerning about the other given one. The rules to reduce a graph depend on the role of a vertex plays in it. Therefore we classify the vertices and define the rules according to each classification.

Focussing on a given *GCDG* we can classify vertices and edges by the amount of information they hold. Figure 2 shows the nine classification types of a *GCDG* . For each type we can define a reduction rule such that there is no loss of information of the network.

Figure 2 Classification of a vertex



Vertex type "empty":

Each occurrence can be removed from the given graph. In fact, there is no reason for having empty vertices in networks.

Vertex type "source" and "sink":

Those vertices remain untouched. In the case of street networks, sources and sinks do not make much sense since one of the main characteristics of street networks is that they are strongly connected digraphs. Sources and sinks do not respect that characteristic. On the other hand, river networks holds those vertex types and therefore they should not been excluded.

Vertex types "dead end", "one way start" and "one way end":

They describe special information about the network. But since they don't describe an intersection, we classify them separately.

Vertex type "one way pass" and "two way pass":

Those vertices do not give additional information to networks. We can remove them and their incident edges by connecting their neighbour vertices with one edge (two edges, resp.). Please note, that–with this rule–we implicitly assume that in the street network, no U-turn is allowed.

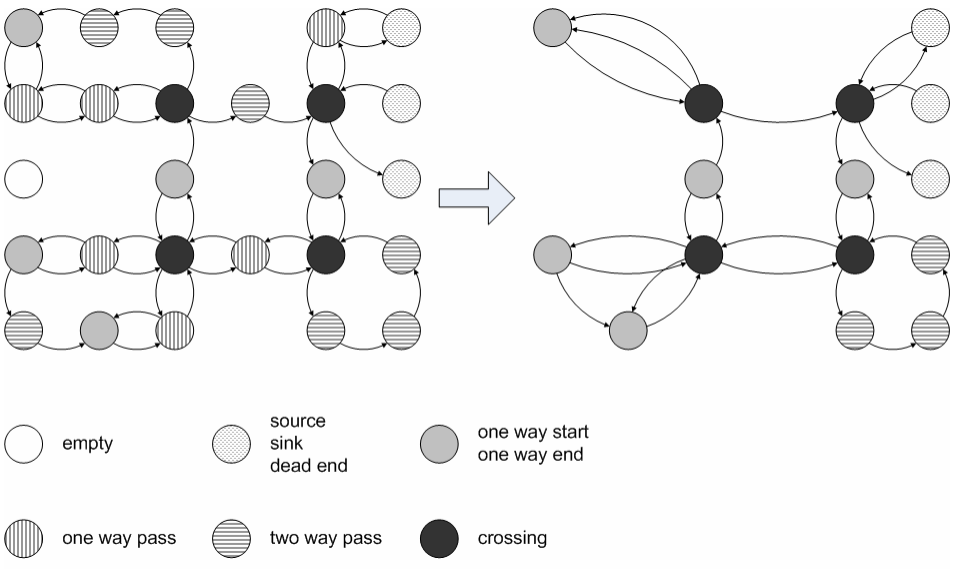
There is one important exception we have to consider. It is possible that in a given graph a path exists which starts and ends at the same vertex and holds only "one way pass" and "two way pass" vertices. If we then follow the rule described in the previous paragraph we would end up with an edge with the same vertex as the source and the sink. Therefore, the rule must not be applied to those vertices.

Vertex type "crossing":

The remaining vertices are crossings. We leave them untouched. For matching two geo-coded digraphs we will concentrate on the "crossing" vertices in the following section.

Figure 3 shows an example of using the above described rules. As we can see in the bottom right loop of the graph, the "one way pass" vertices are not removed, since they are describing a path with the same start and end vertex. The other vertices of these types disappear. The vertex of type "empty" is also removed. Please note, that the upper left part of the graph are reduced in a way that two edges hold the same source (same sink, resp.). It indicates that there are two possibilities to reach the "one way start" vertex.

Figure 3 Example of using single graph reduction rules

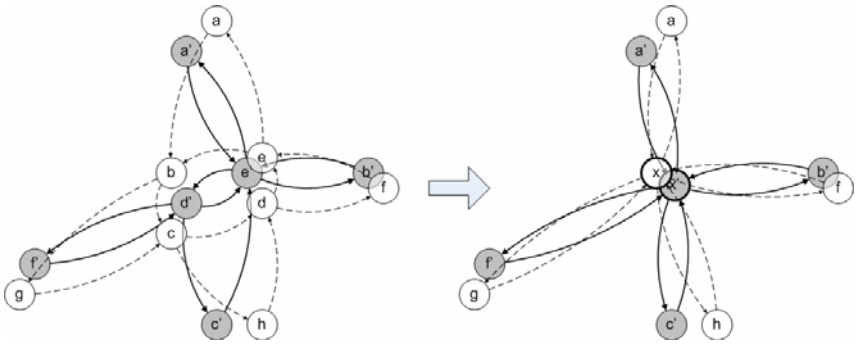


4. Matching Algorithm

The reduction rules described in Section 3.2 applies to a single *GCDG* defining a given (street) network. Those rules do not decrease the amount of information of this network. In other words, if we would remove just one additionally vertex of the resulting reduced network, we would loose essential information. Especially this is valid for the vertices of type "source", "sink", "dead end", "one way start", "one way end" and the remaining "one way pass"/"two way pass" vertices. Therefore, these vertices stay fixed.

The vertices we mostly focus on are of type "crossing". For a single graph they obviously hold essential information. But if we want to match two *GCDG*, we need to reduce the amount of information that we can find a common base. Figure 4 shows the basic idea of reducing two given graphs becoming geo-topologically r-similar (assuming an appropriate value for r). The vertices b, c, d and e of type "crossing" of the white graph are reduced to the vertex x while d' and e' of the grey graph are reduced to vertex x'. If we assume that the border vertices are also of type "crossing", one could say that we can reduce all the vertices of each graph to a single vertex. This is in fact also a common base and the resulting graphs are geo-topologically similar. But we loose too much information (in this case, all information). So, we need to find a common base by which we loose as less information as possible.

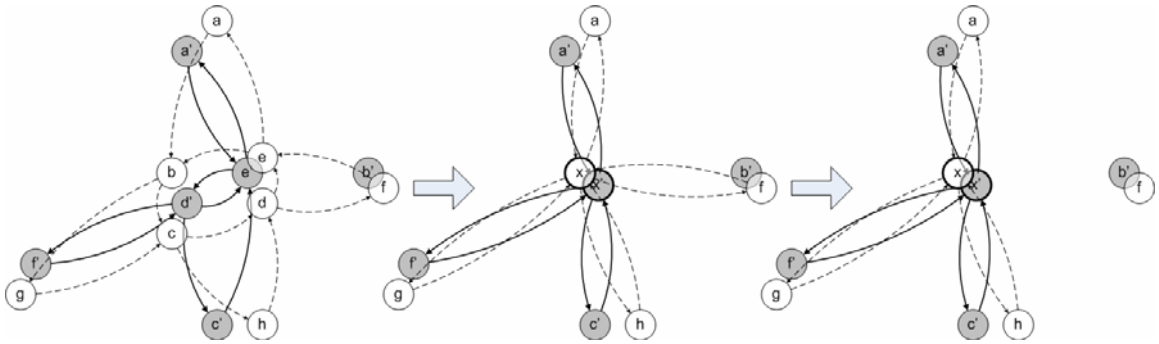
Figure 4 Example of intersection reduction



We need to consider also another fact according to intersection reduction. Typically, the degree of given information differs for two given networks. For example, one street network includes also side roads while the other one just ignores them. Figure 5 shows such an

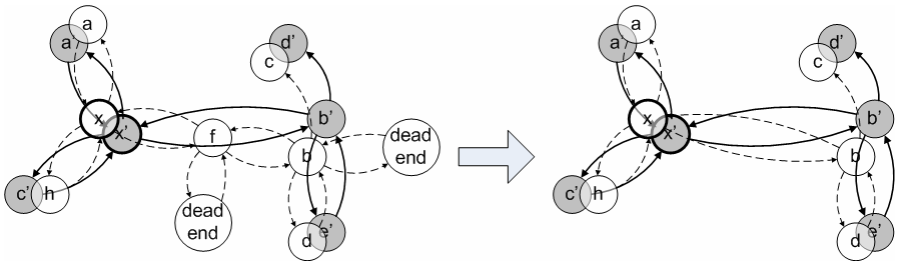
example: In the grey graph the edges (b', e') and (e', b') are not modelled. After the intersection reduction the two graphs are not geo-topologically similar. When we then remove the additional information of the edges (f, x) and (x, f) of the white graph, geo-topological similarity is fulfilled.

Figure 5 Example of intersection reduction with different amount of information



There is at last another fact to be considered. Intersections could be modelled in one graph whereas in another one they do not occur. This is also true for vertices of type “dead end”, “one way start”, “one way end”, “source” and “sink”. Figure 6 shows an example. There is no matching intersection for vertex f of the white graph. Therefore f has to be removed. Also the “dead end” vertices do not have equivalence in the grey graph, so they are removed, too.

Figure 6 Example of removing intersection and dead ends



The core question is how to define an algorithm which first reduces intersections, second removes incident edges of the reduced intersections and third removes vertices holding additional information which cannot be matched to two given $GCDG$ such that they are geo-topologically r-similar with a minimum loss of information of the given graphs. Let us summarize the steps described above:

Given two already classified and reduced geo-coded digraphs $GCDG(V,E)$ and $GCDG(V',E')$ as described in Section 3, then the matching algorithm is defined as follows:

1. Run a *crossing reduction* algorithm, which detects and reduces vertices of type "crossing" of the given graphs.

2. Run an *edge deletion* algorithm, which deletes incident edges of the reduced vertices of one graph that do not have a counterpart in the other graph.

3. Run a *vertex deletion* algorithm, which deletes vertices of any type in one graph that does not have a counterpart in the other one.

The following sections describe the three steps in detail.

4.1 Crossing Reduction Algorithm

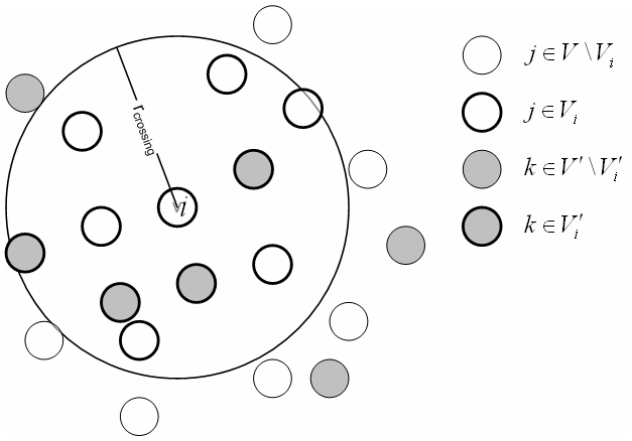
Like Figure 4 already indicates, the crossing reduction algorithm of two given geo-coded digraphs $GCDG(V,E)$ and $GCDG(V',E')$ matches intersections such n vertices of $GCDG(V,E)$ are matched with m vertices of $GCDG(V',E')$. Since we need to check all possible combinations of n vertices in $GCDG(V,E)$ with all combinations of m vertices in $GCDG(V',E')$, the worst case complexity is therefore $O(2^n \cdot 2^m)$ for each single matching. This is far too much and the algorithm would run unfeasibly long for two graphs of reasonable size. But we can take advantage of the fact that the graphs are geo-coded and that we are searching a match which uses as less as possible vertices of type "crossing".

Let us first define an appropriate upper limit for n and m. We do not define that by a fixed number but by a fixed area. This makes more sense since the resolution of the given graphs can differ a lot.

Given two geo-coded digraphs $GCDG(V,E)$ and $GCDG(V',E')$, a vertex of type "crossing" $i \in V$ that is not already crossing reduced and a given radius r_{crossing}, the sets of crossing reduction candidates are defined as $V_i \subseteq V$ and $V_i' \subseteq V'$ such that $\forall j \in V_i : r_{\text{crossing}}^2 \geq (x_j - x_i)^2 + (y_j - y_i)^2$ and $\forall k \in V_i' : r_{\text{crossing}}^2 \geq (x_k - x_i)^2 + (y_k - y_i)^2$ with

$n = |V_i|$, $m = |V_i'|$ and j,k are of type "crossing" and not already crossing reduced. Figure 7 shows a graphical example of the two sets V_i and V_i' with $n = 7$ and $m = 4$.

Figure 7 Example of set of vertices of type "crossing" V_i and V'_i



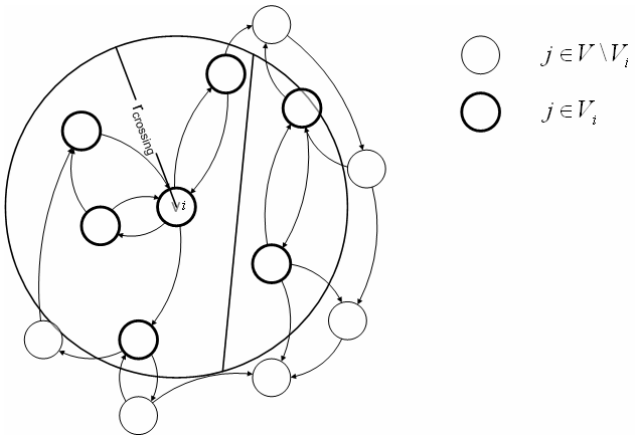
With the knowledge of the two sets V_i and V_i' we can now describe the crossing reduction algorithm:

```
1  For each i ∈ V of type "crossing" do
2    If i is not already crossing reduced, then
3      For n^current = [1,n] do
4        For all sets V_i^current ⊆ V_i with |V_i^current| = n^current do
5          For m^current = [1,m] do
6            For all sets V_i'^current ⊆ V_i' with |V_i'^current| = m^current do
7              If both V_i^current and V_i'^current are connected and
                       V_i^current matches V_i'^current, then
8                Reduce V_i^current and reduce V_i'^current
9                Message "Reduction found with V_i^current and V_i'^current"
10               Goto line 17
11             Done (if)
12           Done (for)
13         Done (for)
14       Done (for)
15     Done (for)
16     Message "No reduction found with initial vertex i"
17   Done (if)
18 Done (for)
```

Note that the messages on line 9 and 16 indicates that we receive a Boolean answer for each initial vertex i. Notice: A reduction is also a matching!

It is necessary to ensure that a current subset $V_i^{current}$ ($V_i'^{current}$, resp.) are connected (see line 7). Otherwise it is possible that the algorithm interprets a vertex set as one intersection even the set holds disjoint parts. See Figure 8 for an example.

Figure 8 Example of set of vertices of type "crossing" V_i with two disjoint parts

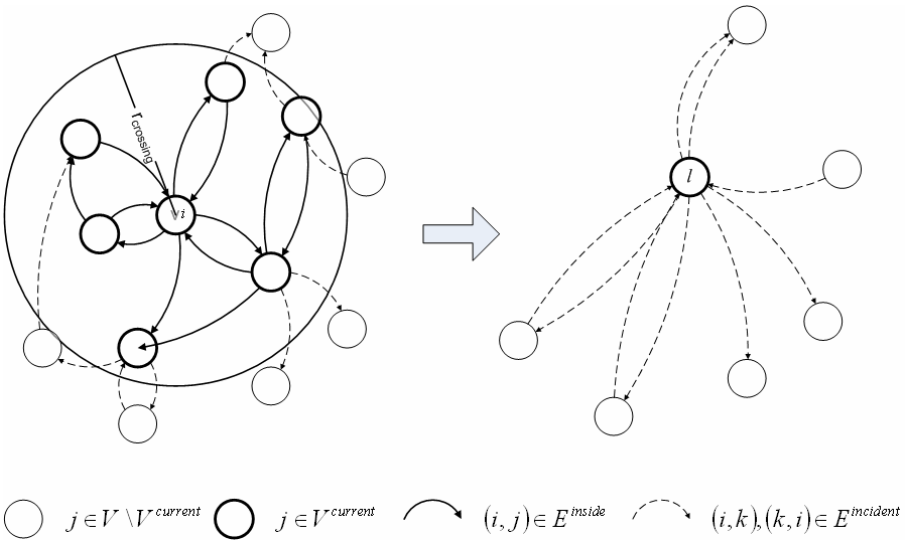


We still need to describe the meaning of "$V_i^{current}$ matches $V_i'^{current}$" (line 7) and "Reduction of a set of vertices" (line 8).

4.1.1 Reduction of $V^{current}$

Given a geo-coded digraph $GCDG(V,E)$ and a set of vertices of type "crossing" $i, j \in V^{current} \subseteq V$ and $k \in V \setminus V^{current}$ with $\{(i,j)\} = E^{inside} \subseteq E$ and $\{(i,k),(k,i)\} = E^{incident} \subseteq E$, then the reduction of $V^{current}$ is a geo-coded digraph $GCDG(V^{reduced}, E^{reduced})$ with $V^{reduced} = \{l\} \cup (V \setminus V^{current})$, $x_l = \sum_{i \in V^{current}} x_i \Big/ |V^{current}|$, $y_l = \sum_{i \in V^{current}} y_i \Big/ |V^{current}|$ and $E^{reduced} = \{(l,k),(k,l)\} \cup (E \setminus (E^{inside} \cup E^{incident}))$. The added vertex l is called "reduction vertex". It is the substitution of the original set $V^{current}$. Figure 9 shows a graphical interpretation of the reduction of $V^{current}$.

Figure 9 Example of a reduction of set $V^{current}$



4.1.2 Matching of $V^{current}$ and $V'^{current}$

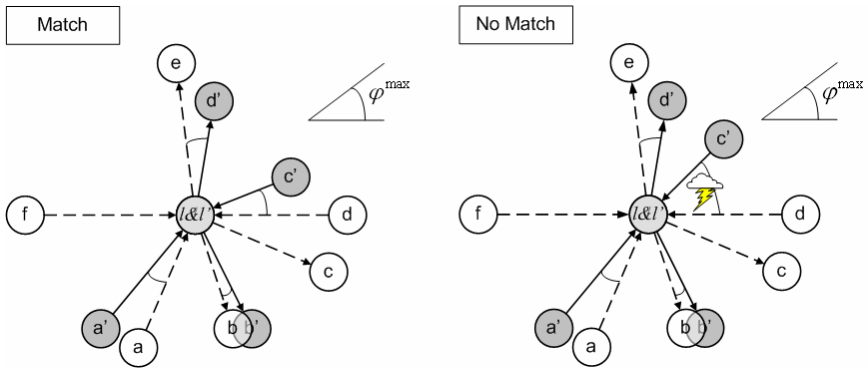
Matching is a binary function. It either returns "true" if the two sets of vertices match or "false" if not.

Given two geo-coded digraphs $GCDG(V,E)$ and $GCDG(V',E')$, two sets of connected vertices of type "crossing" $V^{current} \in V$ and $V'^{current} \in V'$ and a defined angle $\varphi^{\max} = [0,\pi]$, then their reduced graphs $GCDG(V^{reduced}, E^{reduced})$ with "reduction vertex" l and $GCDG(V'^{reduced}, E'^{reduced})$ with "reduction vertex" l' as defined in Section 4.1.1 defines a match of $V^{current}$ and $V'^{current}$, if and only if there is a ordered mapping of all incident edges $E'^{incident}$ of l' with a subset of the incident edges $E^{incident}$ of l with $|E'^{incident}| \leq |E^{incident}|$ according to angle $\varphi^{\max}$. More precisely, given the ordered lists of the incident edges $E^{incident}$ of l and $E'^{incident}$ of l' as defined in Section 2.1, a matching exists if we can build pairs of edges $((l,i),(l',i'))$ and $((i,l),(i',l'))$ with $i \in V^{reduced}$ and $i' \in V'^{reduced}$ such that the angle between the two edges of the pairs are less or equal $\varphi^{\max}$ and the following is fulfilled:

$$\left.\begin{matrix}(l,i)\\(l,i)\\(i,l)\\(i,l)\end{matrix}\right\}\begin{matrix}>^{(a)}\\<^{(b)}\\=^{(c)}\end{matrix}\left\{\begin{matrix}(l,j)\\(j,l)\\(l,j)\\(j,l)\end{matrix}\right\}\wedge\left\{\begin{matrix}((l,i),(l',i'))\\((l,i),(l',i'))\\((i,l),(i',l'))\\((i,l),(i',l'))\end{matrix}\right\}\wedge\left\{\begin{matrix}((l,j),(l',j'))\\((j,l),(j',l'))\\((l,j),(l',j'))\\((j,l),(j',l'))\end{matrix}\right\}\rightarrow\left\{\begin{matrix}(l',i')\\(l',i')\\(i',l')\\(i',l')\end{matrix}\right\}\begin{matrix}>^{(a)}\\<^{(b)}\\=^{(c)}\end{matrix}\left\{\begin{matrix}(l',j')\\(j',l')\\(l',j')\\(j',l')\end{matrix}\right\}$$

Notice that it is not necessary the all edges of $E^{incident}$ belong to a pair while all edges of $E'^{incident}$ are mapped. The graphical interpretation of the above is shown in Figure 10. The drawn angles indicate the mapping. Both mappings respect the order and also all edges of one of the graphs were used. But the mapping on the right side does not match because the angle between edge (d,l) and (c',l') is greater than $\varphi^{\max}$. Please notice that l and l' do not need to have the same position. It is just drawn this way to highlight the angles between two edges.

Figure 10 Example of matching set $V^{current}$ with $V'^{current}$



4.2 Edge Deletion Algorithm

After running the crossing reduction algorithm described in Section 4.1 we receive two resulting geo-coded graphs where some of the vertices of type “crossing” are “reduction vertices”. For each of the “reduction vertices” we also have the information which of the incident edges were not mapped according to the rule described in Section 4.1.2. In principle these edges can be removed from the graphs. Assume that an edge (l,i) incident to the “reduction vertex” l which is not mapped, holds a vertex i which is not a “reduction vertex”, then we cannot remove this edge. Otherwise, the type of vertex i can change (i.e. a “dead end” will be changed into a “source”). Therefore, we remove only those edges (i,j) where both vertices are “reduction vertices” and the edge was not mapped from both sides.

4.3 Vertex Deletion Algorithm

The resulting geo-coded digraphs are still not geo-topologically similar. There are still "crossing" vertices which are not "reduced" ones. The graphs also holds vertices of type "dead end", "one way start", "one way end", "source" and "sink". In addition to that, there are still "loops" as shown in Figure 3. We now look at each type separately. For the following deletion algorithms we will use an area (a circle) of a possible match, called $A(i, r^{match})$, which is defined by a vertex i as the centre and a given radius r^{match}.

Please note, that in the following we use $GCDG^*(V, E)$ and $GCDG^*(V', E')$ as the given geo-coded graphs which are already crossing reduced and edge deleted as described above.

4.3.1 Vertices of type "crossing" that are not already matched

There are various reasons, why those vertices are not already matched (errors in the coordinates or the incident edges could not be match as shown in Figure 10). But there are also two other situations we have to consider:

1. A "crossing" vertex cannot be matched if there is no counterpart in the other graph.
2. A "crossing" vertex cannot be matched if its counterpart is not of type "crossing".

A possible solution to find an appropriate matching can be done by using Dijkstra's shortest path algorithm (Dijkasta, 1959). To use it, we first need to define weights w_{ij} for each edge (i, j) of the given graphs. These are simple defined by the Euclidian distance between the two incident vertices:

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

We will denote the shortest path from vertex i to vertex j as D_{ij}. For both given geo-coded graphs $GCDG^*(V, E)$ and $GCDG^*(V', E')$ we calculate for each pair of already matched "crossing" vertices i and j (i' and j', resp.) the shortest path D_{ij} ($D_{i'j'}$, resp.).

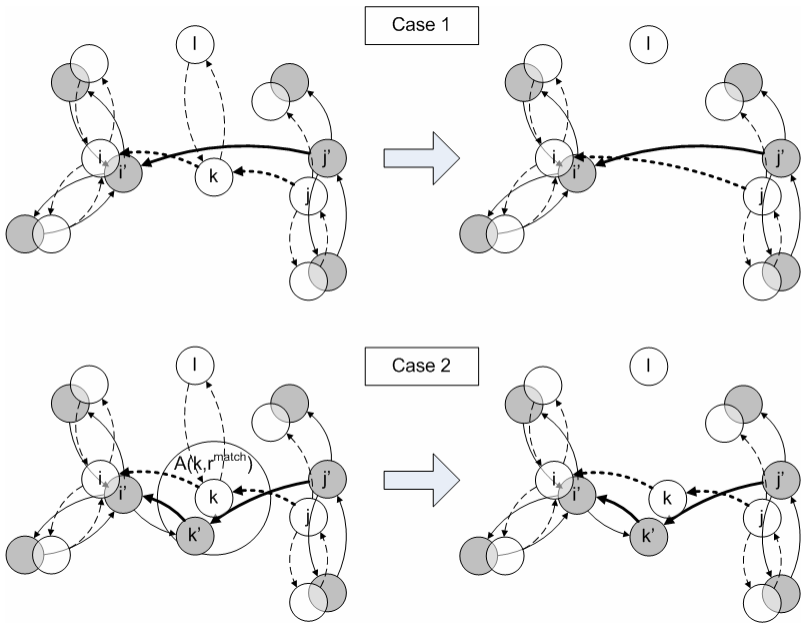
Given a "crossing" vertex k that is not already matched and a path D_{ij} with $k \in D_{ij}$, $(s,k),(k,t) \in D_{ij}$ and a path $D_{i'j'}$ as the counterpart of D_{ij}.

1. If there is no vertex $k' \in D_{i'j'}$, delete all incident edges of k and set edge (s,t). Also set an edge (t,s), if there is also a path D_{ji} and $D_{j'i'}$. Also update the types of the neighbour vertices of k.

2. If there is a vertex $k' \in D_{i'j'}$ and k' lies inside the area $A(k, r^{match})$, then delete all incident edges which do not belong to path D_{ij}. Also update the neighbour vertices of k and also k (it is not a "crossing" vertex anymore). Mark k and k' as matched. If k' does not lie inside $A(k, r^{match})$, construct the same result as in 1., which means that we delete k and k'.

Figure 11 shows examples of the two situations described above. The bold edges highlight the paths D_{ij}, $D_{i'j'}$ resp.

Figure 11 Example of two cases of a "crossing" vertex that is not matched already



Let us at last focus on the vertices which their type was updated (vertex l in Figure 11). It is possible that after this step, l is part of a disjoint sub graph of $GCDG^*(V,E)$. This is true, if l was either part of a path from a "dead end" to vertex k or part of a loop with k as the only vertex of type "crossing". Therefore, we just delete the whole disjoint sub graph.

Now, no "crossing" vertices exist anymore which are not matched. This helps a lot for the following steps:

4.3.2 Vertices of type "dead end"

A vertex $i \in V$ of type "dead end" has a shortest path D_{ij} to a vertex j of type "crossing". If there is a vertex $i' \in V'$ inside the area $A\left(i, r^{match}\right)$ which has a path $D_{i'j'}$ to j's counterpart j', the whole path D_{ij} can be matched with $D_{i'j'}$. It is possible that i' is not a "dead end" vertex. In this case, we are allowed to delete the excessive incident edges of i'.

4.3.3 Vertices inside a loop

Because of the step described in Section 4.3.1 vertices $i \in V$ inside a loop with the single "crossing" vertex j can only be matched if there is also a loop in $GCDG^{*}\left(V', E'\right)$ with j's counterpart j'.

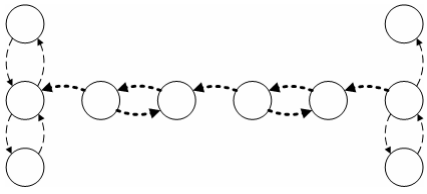
4.3.4 Vertices of type "source" or "sink"

For sources and sinks we can follow the same rule as described in Section 4.3.2. We only need to find one path D_{ij} (D_{ji} for sinks) for a "source" / "sink" vertex $i \in V$ to a "crossing" vertex j. But according to street networks, it does not make much sense that a given graph holds sources or sinks. In such a case, the user should check why such vertices exist.

4.3.5 Vertices of type "one way start" or "one way end"

After the steps of Sections 4.3.1 to 4.3.3 almost all vertices of type "one way start" or "one way end" are matched or deleted. If there are still some left then they are part of a chain of such vertices. An example is shown in Figure 12.

Figure 12 Example of a chain of "one way start" and "one way end" vertices



It is again not completely clear why such situations are modelled in a graph. Nevertheless, we can again match the whole path D_{ij} from one "crossing" vertex to the other one with its counterpart $D_{i'j'}$ if a path like this exists. Otherwise we delete the path D_{ij}.

4.4 Summary

In principle the whole matching algorithm above described for two geo-coded digraphs is feasible. But there is one important fact about it we have to point out. The whole algorithm is based on three parameters:

- r_{crossing}, that defines the maximum area where the crossing reduction algorithm tries to find a vertex sets which can be reduced,
- $\varphi^{\max}$, that defines the maximum angle between two edges which is allowed for a matching,
- and r^{match}, that defines the maximum distance for a one-to-one matching in the vertex deletion algorithm.

If at least one of these values is set too small, the matching algorithm almost never finds an appropriate matching. On the other hand, we could also set r_{crossing} and r^{match} to infinity, but then, the complexity of the algorithm increases too much.

Let us assume that the three parameters are set to appropriate values, then the two resulting geo-coded digraph do not have to be geo-topologically similar. Figure 11, case 2 shows an example: The grey graph holds the edge (i', k') but its counterpart (i, k) does not exist. Of course it is now very easy to reduce one graph such that they fulfil geo-topological similarity but it is not necessary. Since we already found the matching parts of the graph, this last reduction is only cosmetics.

5. Test Cases

To test the above described matching algorithm a set of manual generated geo-coded digraphs are used. The graphs are set in a defined area of 50 meter width and 30 meter height. Each graph holds between 2 and 15 vertices, 1 and 50 edges resp. To respect the pre-conditions described in Section 3.1 we create a pair of graphs on a base of one randomly generated digraph by adding / removing vertices and edges (still respecting the pre-conditions), by adding noise to the coordinates of the vertices and to expand vertices of type "crossing" by replacing them with a set of vertices holding the topology of the replaced one. The reason for using only such small graphs are that we can define $r_{\text{crossing}} \to \infty$ and $r^{match} \to \infty$ without worrying about computation time. The following subsections will present the success rate of the matching algorithm based one different classes of pairs of geo-coded digraphs.

Confusion Matrix

The success rate is measured with the "precision" and "recall" calculation based on the confusion matrix (see Witten and Frank, 2000). The confusion matrix is defined as follow (see Figure 13):

- true positives (TP): number of class members classified as class members
- true negatives (TN): number of class non-members classified as non-members
- false positives (FP): number of class non-members classified as class members
- false negatives (FN): number of class members classified as class non-members

Figure 13 Confusion matrix

		predicted class	
		yes	no
actual class	yes	true positive	false negative
	no	false positive	true negative

For a set of vertices (representing an intersection or just single vertex like a “dead end”) of $GCDG(V, E)$ and a set vertices of $GCDG(V', E')$, a matching is

- true positives (TP) if the two sets are representing the same information and the matching algorithm finds the match.
- true negatives (TN) if the two sets are representing different information and the matching algorithm does not find the match.
- false positives (FP) if the two sets are representing different information but the matching algorithm finds a match.
- false negatives (FN) if the two sets are representing the same information but the matching algorithm does not find a match.

Therefore, precision is the fraction of the number of correct matches to the total number of matches of the algorithm:

$$precision = \frac{TP}{TP + FP}$$

Recall is defined as the fraction of the number of correct matches to the total number of intersections which have to be matched:

$$recall = \frac{TP}{TP + FN}$$

If precision and recall are equal to 1, the algorithm produces a perfect match.

5.1 Matching two geo-topological similar GCDGs

The two graphs are created out of a given graph. To one of the two graphs we add a certain amount of noise with upper limit of $r^{noise} = [0,50]$ meters to the coordinates of its vertices. Therefore, a perfect match holds only one-to-one matches of the vertices.

- If the graphs are geo-topologically equal, the algorithm always finds all matching ($precision = 1$ and $recall = 1$).
- For a small amount of noise and $\varphi^{\max}$ greater than zero (around $\pi/4$) the algorithm finds almost all matching intersections($precision \approx 0.99$ and $recall \approx 0.95$). If the maximum angle is chosen to small then precision and recall decreases rapidly.

- If the noise is chosen too large ($r^{noise} \approx 40$) with large $\varphi^{\max}$ then the algorithm often matches the wrong pair of vertices and sometimes it also produces "reduction vertices". Therefore, the $precision \approx 0.0$ and recall $recall \approx 0.0$.

These test cases show that with some amount of noise in the coordinates the algorithm still produces a good matching. But if the noise is getting too large (i.e. created by measurement errors on the given network) mismatching can occur. The different runs also shows that an appropriate maximum angle $\varphi^{\max}$ lies approximately between $\pi/6$ and $\pi/3$.

5.2 Matching two GCDGs while one holds detailed intersection information

The two graphs are created out of a given graph. The "crossing" vertices of one graph are then extended with more than one vertex such that some of the incident edges a "crossing" vertex are divided into two pieces with an additional vertex. Then these vertices were randomly connected by additional edges. Therefore an appropriate match of two intersections is always a one-to-many match.

- For $\varphi^{\max} \approx [\pi/6, \pi/3]$ the algorithm finds most of the matching ($precision \approx 0.99$ and $recall \approx 0.9$). Mismatching occur only because only part of the expanded "crossing" vertices are matched to its single counterpart.

- If the angle set $\varphi^{\max}$ too small or too large the precision and the recall decreases again. But even $\varphi^{\max} = \pi$ the algorithm still finds correct matched intersections ($precision \approx 0.3$ and $recall \approx 0.25$).

5.3 Matching two GCDGs while both holds detailed but different intersection information

This setup is similar to the one above, but this time we also extend the "crossing" vertices of the other graph by splitting up another subset of the incident edges of the "crossing" vertex.

- For $\varphi^{\max} \approx [\pi/6, \pi/3]$ the algorithm slightly less matching intersections ($precision \approx 0.94$ and $recall \approx 0.89$). The reason is the same as above.

- $\varphi^{\max}$ again should not be too small or too large.

5.4 Matching two GCDGs while one holds additional vertices and edges

The two graphs are created out of a given graph. One stays the same while the other one holds additional vertices and edges. Therefore, one graph is a subset of the other one. The maximum angle $\varphi^{\max}$ is set between $\pi/6$ and $\pi/3$.

- The algorithm almost always finds a perfect match ($precision \approx 0.99$ and $recall \approx 0.99$). This shows that the edge deletion algorithm and the vertex deletion algorithm produce the expected results for those test cases.

5.5 Combinations

The above described four different test sets can be combined to produce "more difficult" pairs of graphs. We again keep the maximum angle $\varphi^{\max}$ between $\pi/6$ and $\pi/3$.

Combination of Section 5.2 and Section 5.4:

The algorithm still works surprisingly well ($precision \approx 0.96$ and $recall \approx 0.88$), because the crossing reduction algorithm finds again many of the vertex set which has to be reduced. Then the vertex and edge deletion algorithms can work on similar pairs of graphs like described in Section 5.4.

Combination of Section 5.3 and Section 5.4:

Compared to the above combination the algorithm produces more mismatches ($precision \approx 0.92$ and $recall \approx 0.87$). They occur because the crossing reduction algorithm sometimes reduces a vertex set holding a vertex which was added like described in Section 5.4. Those vertices then cannot be deleted anymore (they are already replaced by a "reduction vertex").

Combination of Section 5.1 with Section 5.2, Section 5.3 or Section 5.4:

We have already seen that too much noise let the algorithm fall apart. Therefore, we will define the range of noise as $r^{noise} =]0,30]$. Of course we do not set $r^{noise} = 0$.

- If r^{noise} is small the algorithm still produces feasible precision and recall values ($precision \approx [0.8,1]$ and $recall \approx [0.7,0.98]$). This is quite expecting since the different parts of the algorithm do not react too much one a small amount of noise.

- If r^{noise} is increased to values between 20 and 30 meter the outcome of the algorithm varies a lot for different pairs of input graphs ($precision \approx [0.0,0.9]$ and $recall \approx [0.0,0.9]$). It is not that easy to see why this happen. One possible answer is that sometimes the noise changes the geo-topology of a graph while sometimes the geo-topology remains the same.

To sum up, the different test runs show that the algorithm reacts quite sensitive on noise. Also a feasible maximum angle $\varphi^{\max}$ has to be defined. On the other hand it shows that the crossing reduction algorithm finds at least an appropriate amount of correct reductions. The edge and vertex deletion algorithms are quite stable.

But we have to point out that all these tests are done on really small graphs. We also do not know yet how the algorithm reacts on the parameters r_{crossing} and r^{match} when they are set to a finite size.

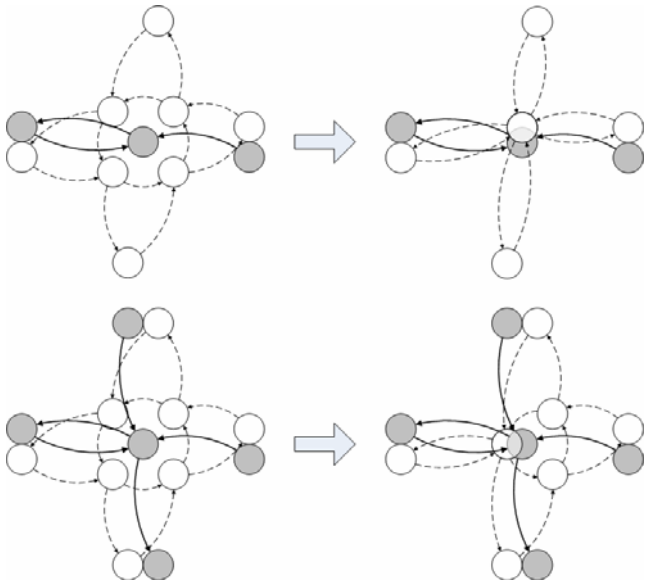
6. Summary and Outlook

Matching two geo-coded digraphs is clearly not a trivial task. Nevertheless, the graph matching algorithm presented in this paper shows some promising results. Even more, the algorithm do not use any additional information available in different networks except of the position of the vertices. This makes the algorithm usable to any kind of input networks.

But we have to point out that there is much more analysis to do. The graph matching algorithm is only tested against small (randomly generated) graphs. It is now of interests to see how the algorithm behaves on “real” and larger networks.

Last but not least it is not completely clear if two geo-coded digraphs can be perfectly matched. It is also to discuss if “crossing reduction” is the right way to go to match two networks. Figure 14 shows how the reduction algorithm interprets intersection differently depending on the given networks. The upper reduction is done assuming that the four white vertices describe one intersection (i.e. a roundabout). The grey graph holds less information and the intersection is modelled as a “one way end” vertex. In the lower case the grey graph has more information but still modelling the same intersection as one vertex. The crossing reduction algorithm now interprets the four white vertices as three separate intersections even though the white graph has not changed.

Figure 14 Different interpretation of an intersection of the crossing reduction algorithm



7. References

Christmas, W.J. (1995). Structural matching in computer vision using probabilistic reasoning. PhD thesis, University of Surrey, Guildford, U.K.

Christmas, W.J., J. Kittler and M. Petrou (1995). Structural matching in computer vision using probabilistic relaxation. In *Proceedings of Pattern Analysis and Machine Intelligence, volume 17*.

De Palma, A., F. Marchal and Y. Nesterov (1997). METROPOLIS: A Modular System for Dynamic Traffic Simulation. Transportation Research Record

Dijkstra, E. W. (1959). A Note on Two Problems in Connection with Graphs. *Numerische Math.*, **1**, pages 269-271.

Fitch, A. J., A. Kadyrov, W.J. Christmas and J. Kittler (2002). Fast Exhaustive Robust Matching. In *Proceedings of ICPR 2002, volume III*, pages 903-906

Luks, E. M. (1982). Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time. *J. Comput. System Sci*, **25**, pages 42-49.

Pendyala, M. (2002). Development of GIS-based conflation tools for data integration and matching. Final report for Research Center of Florida department of transportation, Tallahassee, FL.

Skiena, S. (1990). Graph Isomorphism. Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. *Reading, MA*, Addison-Wesley, pages 181-187.

Sloane, N. J. A. and S. Plouffe (1995) The Encyclopedia of Integer Sequences. San Diego, CA, Academic Press.

Waldner, U. (2005). Automatic Matching of large, detailed Street Networks - An algorithm with curve and attribute affinity and its evaluation. In *Proceedings of Swiss Transport Research Conference (STRC)*, Monte Verita, Switzerland. See www.strc.ch (Accessed Feb. 2005).

West, D. B. (2000). Introduction to Graph Theory. *2nd ed. Englewood Cliffs*, Prentice-Hall, NJ.

Witten, I. H. and E. Frank (2000). Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. *Academic Press*, pages 137-147, San Diego, CA.