PATH DIAGRAMS

CONFIGURATIONAL DESCRIPTIONS FROM GIS DATA, AN ALGORITHM AND ITS IMPLICATIONS.

PABLO MIRANDA CARRANZA; ANN LEGEBY; DANIEL KOCH

ABSTRACT

This paper proposes and explains a type of diagrams which share some of the characteristics distinctive of axial maps, while also differing substantially in some other aspects. A main discussion in Space Syntax is the capacity of different diagrammatic representations to describe relevant morphological characteristics of space: convex maps or visibility graphs at the scale of buildings; axial maps and other topological representations of streets or roads at urban and geographic scales. The relevance of these representations is often assessed through the correlation of the different metrics they afford with empirical observations of real phenomena, such as the flow, circulation and presence of people in space. This has led to an instrumentalist understanding of representations and models through their capacity to produce measurements that may be empirically correlated, relegating other aspects to a secondary plane. In their “Space Syntax” paper from 1976, Hillier et al. outlined a research programme based on the proposition of morphic languages and their syntaxes; the collection of maps and diagrams proposed by Hillier and Hanson years later in “the Social Logic of Space,” including the axial map, carried further this programme of a science of diagrams and “morphic languages,” rather than one focused on metrics and their empirical correlations. This paper wants to reconsider some of these original characteristics of the representations of spatial configurations, through the proposition of a diagram and an algorithm to generate it. This diagram has been developed for its use in the comparison and classification of spatial configurations. Using data available from Open Street Map, the algorithm presented here builds a graph in which its vertices consist of straight paths on the original graph, and edges represent crossings or adjacencies of these straight paths. The process involves the implicit construction of what is known in graph theory as the line graph and a set of operations so as to derive the final graph from it. This graph is referred to in the paper as the “straight path graph,” as it consists of paths, both in their graph theoretical sense of a sequence of non-repeated vertices, and in their more general sense as circulation paths on a map. The insistence on its qualities as a diagram is the result of a need to describe more concisely spatial organisations in order to map, compare and classify them, either by hand or through the use of diverse automatic techniques and algorithms, rather than its use in the production and correlation of metrics with empirical observations.
1. INTRODUCTION

The subject of this paper is principally technical: data representations of graphs that describe circulation configurations so they can be used in pattern recognition and clustering algorithms. But as all technical systems these also articulate schemas and concepts that impose biases and create and reduce potentials, and through these, organise and reproduce different forms of discourse. Any following technical description should be read with this in mind: not only as the proposition on an algorithm and its potential uses, but also as a critique of the instrumentalist bias that metrics and empirical correlation impose on computational descriptions of space, in detriment of their configurational and syntactical characteristics.

As a first step in considering these graph representations we should clarify the epistemological framework that has given rise to them. According to Gilbert Simondon knowledge can be conceived as a problem of representation: in an eidetic tradition the object of knowledge is not the true and actual object, but a double in the form of ideas that define its structure, ideas which exit before they have even been thought. In this tradition knowledge is neither formed nor constructed by the subject; there is no genesis of knowledge, only the discovery of the real by the mind through ideas. On the contrary, operational knowledge considers the possibility of constructing its object; according to Simondon (Simondon 1958, Simondon, Malaspina et al. 2017) this object is produced through representations resulting from manipulable elements, produced in the same way artisans constructs their objects: they are the result of techne and in a broad sense technological. The concept, rather than the idea, is the instrument of operational knowledge, the result of a process of assembly characterised by abstraction and generalisation. Knowledge is then contingent to its genesis, rather than being prior to all human gesture. For operational knowledge, the real does not precede the operation of knowledge; it comes after it.

The algorithm and diagram presented here have been conceived within this aposterioristic view; their intention is not to capture and represent an underlying reality, but to function as instruments that conceptualise notions such as configuration, pattern or layout. It is important to remind also that the development of any algorithm or computer program is always contingent to the material constrains and theories determining its appearance: the organisation of the machine in which it is implemented and run, the programming language in which it is conceptualised and written, the original purpose of the data employed and the technical and political conditions for its storage and distribution, from intellectual property rights to the public or private ownership of the server. All these account for the characteristics, form and structure of the results. In a broader sense the algorithm presented here can be then seen as a diagram in the sense proposed by Deleuze and Guattari or Gilles Châtelet (Deleuze and Guattari 1987, Châtelet 2000). In Deleuze’s and Guattari’s formulation, diagrams do not mean, but rather they organise and distribute, constituting, literally in the case of algorithms, abstract machines.
This may perhaps sound like a rather excessive theoretical framing of a simple technical proposition, but for it to become meaningful and to be critically assessed it is necessary to consider its intentions and make explicit its limitations. This is far from anecdotic in the field to which this symposium belongs. Hillier, Leaman, Stanstall and Bedford based much of their explanations in the original “Space Syntax” paper (Hillier, Leaman et al. 1976) on epistemological grounds. Space Syntax was thus proposed as a way of remedying the lack of mathematical theories of among other things settlement patterns in the ‘sciences of the artificial,’ a deficiency explained as the result of the operational knowledge which dominated them. Space Syntax was in contrast proposed as a belief in an aprioristic mathematical order inherent in its objects of study. It is important thus to first emphasise the different epistemological consequences of grounding research on empirical, idealist or operational premises, as these will have as their different objects of knowledge the sensed data of real phenomena, the underlying forms of this reality, or the contingent instruments and concepts organising its understanding.

Without getting too much into fundamentally epistemological polemics, the work in this paper is based on the believe that in order to develop representations that are suited to be effectively treated by computers, it is a pragmatic imperative to begin from diagrammatic representations rather than from a real phenomenon. To take an operational standpoint that considers abstract entities such as algorithms and data structures and their formal limitations as well as those of their material implementations will circumvent the problems of translating into digital procedures ideal models that may not lend themselves easily to algorithmic representation. This is the case of many concepts central to Space Syntax such as convex maps or axial lines, which are hard to effectively, strictly and unambiguously formulate as algorithms.

One of the arguments in 1976 to propose a Space Syntax was to consider the formal order of systems such as cities in relation to their potential use in artificial intelligence; rather than beginning from existing mathematical models, the original 1976 paper suggested to mathematise intuitive formal principles. The direction of the small example presented here is then the opposite one; besides the practical use of these models in pattern recognition and clustering algorithms, it is the hope that this inversion of direction will contribute to a general critique of models and representations, to problematise and contribute to the nature of the “morphic languages” that were the contribution of early Space Syntax research.

2. A FEW THOUGHTS ABOUT SPACE SYNTAX DIAGRAMS
The availability of GIS data has given rise to a problem in Space Syntax: how to translate the types of topological descriptions of roads and paths used in these models into representations relevant to the field (Dalton, Peponis et al. 2003, Turner 2007). The general approach to validate these translations has stressed correlations between syntactic measures from different types of representation (axial lines, centre lines…) and empirical observations of uses of space, such as the pedestrian traffic through a network of streets, the circulation through a building, or the concentration of people in
squares, rooms or locations within them. These measures, often based in standard network algorithms such as the calculations of shortest paths depending on different cost functions --topological, angular, or metric distances--, and associated measures such as betweenness and closeness centrality, relate often to specific real phenomena of human occupation for which the topological configuration of space is an important factor. In this sense an algorithm like Dijkstra’s or a procedure like a breadth first search may either correspond to cognitive process performed by subjects moving through the network or otherwise to the accumulative effect of the choices these subjects make. The stress on measurement through the network, rather than on the actual conceptual components that make up its topology (the entities that the vertices in the graph represent and the type of relations that edges define), has shifted attention from the syntax of spatial configurations and the schemata of the “morphic languages” that may represent them (as their a priori conditions or as a posteriori conceptualisations) to the phenomena they regulate and organise. This correlation is certainly fundamental to a theory that wants to move beyond the formalisms of its representations to become both explanatory of measurable phenomena (pedestrian circulation, social interactions…), and technical, in the sense of being able to affect these phenomena. Both formalism and empirical considerations were part of the research programme of Space Syntax as it was formulated more than 40 years ago (Hillier, Leaman et al. 1976), but the excessive prominence given to measures and correlations has shifted attention away from the conceptual differences between these models and from the possibility of their critique outside these narrow evaluation criteria.

The algorithm and diagram presented in this paper want to return to some of the original concerns of Space Syntax with producing syntactic models linked to the research carried out in the field of artificial intelligence. Quantities and measures may distinguish the performance of spatial systems, but in order to characterise their specific configurational aspects, rather than their effects, it becomes necessary to describe these configurations clearly. Also, any computational processes that will compare and identify recurring patterns, similarities and differences, will be driven by an economy of the elements used to represent configurations: the least elements and relations used, the faster the processing and the larger the scalability of the representation. These elements and relations should be meaningful, in the sense that they should be able to index a real phenomenon, but also reductive, in the sense that they should disregard as much as possible all aspects irrelevant to the basis of their comparison. In this reduction, many details of the description, which may be important in the fine-tuning of measurements in relation to empirical correlations, will be filtered out and voluntarily disregarded as sources of noise in the description.

The archetypical diagrams of Space Syntax such as the axial map or the convex map (Hillier and Hanson 1984) are, in view of the above, ideal examples of this meaningful reductive sparseness: many of the problems associates with them when it comes to the limitations of their descriptions of street patterns and circulations (Ratti 2004), or correlations with actual empirical observations (Turner and Penn 1999) do not consider that this is the price paid for the simplicity of the model and the sparseness of the descriptions it allows, a price that may well be worth to pay in exchange for their
capacity to describe, in a very condensed manner, the most important aspects of street patterns or spatial configurations. The economy of their descriptions is not only useful in their application in pattern recognition or other algorithms, but an important feature if they are going to be manipulated as part of a design process. A second problem of axial maps is the difficulty in describing what is it exactly that they capture, what aspect of reality do they really represent: a direct translation of streets as a diagrammatic entity, directions of walk, visual and cognitive entities, or affordances for walkability. This essential meaning of axial lines is important if one considers that they may relate to an a priori quality of space, but, seen instead as a diagram, the multitude of possible interpretations only accounts for their potential to structure and articulate a varied number of concepts. These observations are also relevant in relation to the early postulation of the research task of Space Syntax, which had as an objective “not to say why people deploy themselves in space, but to offer a theory of patterns” (Hillier, Leaman et al. 1976). Unfortunately, a final characteristic of axial lines is an impediment for their use through algorithms: even if the first formula for their generation in “The Social Logic of Space” by Hiller and Hanson (Hillier and Hanson 1984) is general enough to be reproduce by hand, and despite the fact that there have been a number of propositions to generate axial lines algorithmically (Peponis, Wineman et al. 1998, Batty and Rana 2004, Turner, Penn et al. 2005), in practice it is hard to use these procedures fully automatically for kinds of available data, since there always exist particularities related to the type of the spatial entities to generate axial lines from (traditional urban structures, modernistic suburbs, road networks, landscapes…) that produce differences that are hard to account for in a generic algorithm.

The type of diagram presented here uses some of the characteristics of axial lines as a heuristic to process GIS data into simple descriptions of configurations. It particularly uses Open Street Map data of circulation paths, accessed directly from software through queries in the Overpass QL language. The resulting descriptions have been used to compare configurations using the graph edit distance as proposed by Ruth Conroy Dalton and Ciler Kirsan for building layouts (Dalton and Kirsan 2005, Conroy Dalton and Kirsan 2008), by implementing the heuristics described by Riesen, Neuhaus and Bunke(Riesen, Neuhaus et al. 2007). We won’t discuss the details of this comparison algorithm or the hierarchical clustering used for the classification of street patterns in this paper, but instead those of the construction of the graph used in these algorithms.

3. THE COMPARISON OF CIRCULATION GRAPHS: TYPES AND POPULATIONS.

The use of urban typologies, the classification of plans of buildings and their effect on urban space is part of the common practice of urban analysis and planning. The study of the diversity or homogeneity of building and urban types, the interest in preserving, challenging, complementing or erasing tower blocks, terraced houses or perimeter blocks, identified through the geometry of their building footprint, are customary in most planning processes. Typology has played an important role in architecture; Giulio Carlo Argan (Argan 1963) provided one of the best known definitions, largely based on Antoine- Chrysostome Quatremère de Quincy’s first description of type from 1825. According to Argan a type arises from the existence of a series of buildings having between them an obvious formal and functional analogy; In the process of comparing and superimposing individual
forms so as to determine their type, particular characteristics of each individual building are eliminated and only those remain which are common to every unit of the series. Type is thus formed through a process of reducing a complex of formal variants to a common root form. This form contains the possibility of its infinite variation and modification into instances of the type. Typologies are not just simply the result of classification or statistical processes, but they are carried out in search of a schema or outline of a form.

Automatic processes as those available to artificial intelligence depend instead on classification and statistical calculations. Rather than typological, their working is closer to what biologist Ernst Mayr described as “population thinking,” in which unique entities are classified into species as a consequence of statistical differences and commonalities, rather than as the result of exhibiting ‘typical’ traits (Mayr 1976). This is clearly the case in the example presented in this paper, in which a hierarchical clustering algorithm is used to classify a population of pedestrian circulation networks through a dendrogram (a typical diagram used in biology and genetics).

The classification of pedestrian networks or street patterns using an automatic process shown here can be divided into two different main problems: first, the development of convenient representations and models of configurations that can be easily derived from existing data, and which capture relevant and comparable aspects; second, the application of algorithms to recognise, identify and classify similitudes and differences between configurations. This paper explains mainly an approach to the first problem, by describing how to produce adequate graph representations from existing GIS data; but before describing this algorithm, we will briefly introduce the context in which it has been applied by shortly addressing the strategies used for the second problem. There are a number of different techniques to compare graphs: typical methods involve the embedding in a space of vector descriptions of graphs, so similar structures are close together and dissimilar ones are far apart (Wilson, Hancock et al. 2005). Another method is to use edit distances, a standard measure employed in comparing strings in computational linguistics (Wagner and Fischer 1974), by calculating how much one graph would need to be modified to match another. Put simply, the edit distance consists of assigning costs to the different operations (substitution, addition or deletion of vertices and edges) needed to transform one graph into the other, and then to choose the transformations that imply a lowest cost. This cost would indicate how much the two graphs resemble each other (low cost would imply similarity, high cost difference). The edit distance, albeit in its strict version, was already used by Ruth Conroy Dalton and Ciler Kirsan for the comparison of layouts of Greek and Turkish houses in Cyprus (Conroy Dalton and Kirsan 2008). While they demonstrated the usefulness of the method, their strict implementation of the edit distance makes its extension impractical for any graph but the absolutely most simple ones: strict implementations of the edit distance for graphs have exponential complexity, making the comparison of non-trivial graphs practically intractable. On the other hand, a number of methods have been proposed that use a set of heuristics that approximate the exact calculations, but with a much lower computational cost (Riesen, Neuhaus et al. 2007). This is the
method employed in the comparisons and classifications of graphs resulting from the algorithm described in this paper.

4. THE STRAIGHT PATH GRAPH

We have called the type of the diagram resulting from the algorithm described here the ‘straight path graph.’ There are two main reasons for this name: first, the GIS data it operates on is pedestrian circulation network data from Open Street Map. The deadpan definition of the elements of the graph as “straight paths” avoids inclinations to make grand claims about what the diagram represents. Second, ‘straight path’ corresponds accurately with the graph theoretical description of the entities used as vertices in the graph.

A path in a graph is defined as a walk, that is, a finite non-null sequence whose terms are alternately vertices and edges beginning on a vertex and ending on another, in which all vertices are distinct (no vertices are visited more than once) (Bondy 1976). In this case the paths are generated over a graph G constructed from OSM queries, in which points with given longitudes and latitudes are the vertices, and the line segments connecting them are the edges. The straight paths are then walks in this graph that don’t visit a vertex more than once and in which a relation defined as the “angle” between any two adjacent edges in the path is larger than a certain value $\alpha$. A further constraint for the paths is that the sum of all of these “angles” is as small as possible. This will guarantee that the paths are as “straight” as they can be, and reduces the arbitrariness of the straight path graph (SPG) for a given graph G (more than one SPG may be possible from the same G if there are many identical “angle” relations between edges; uniqueness can be enforced through a strict ordering using some other factor).

The resulting straight paths can share the same vertex, but never the same edge (different paths may cross or begin and end at the same vertex in G). The straight path graph SPG has these paths in G as its vertices, and the shared vertices in G as its edges.
Figure 1: From left to right and top to bottom: The original circulation graph from OSM; the straight paths generated; the straight path graph; the straight path graph with global depth of the vertices represented through line thicknesses.
5. A DESCRIPTION OF THE ALGORITHM

Before describing the implementation, we could look at a general description of the algorithm. This takes as its input a graph G of the type described above, and first derives its corresponding line graph or L(G), also known as the edge-to-vertex dual. The vertices of L(G) correspond to the edges of G, and two vertices of L(G) are adjacent whenever the corresponding edges of G are adjacent (Harary and Norman 1960). The L(G) allows then to have description of pairwise relations between edges in G: each of these pairwise relations (the edges of L(G)) can be assigned a cost (its “angle”) depending of its two adjacent edges in G. This cost may be a simple linear function of the actual angle between the edges (the case in the implemented version) or any other description of lack of straightness between edges (this could be also a function of the length of the edges, for example, thus the quotation marks around “angle”).

The algorithm used to produce the straight path graph using L(G), resembles closely Kruskal’s algorithm for calculating minimum spanning trees (Kruskal 1956, Cormen 2009). In this case paths are seen as a special type of trees with either a single branch or two branches, depending of which vertex we consider as the root. Kruskal is a greedy algorithm which proceeds by building a forest F (a set of trees) in which each tree is a subset of the minimum spanning tree of the graph. The forest F will initially consist of trivial trees made of one vertex and include all vertices in the graph. All edges in the graph will then be considered sequentially from the shortest to the largest: if the edge connects two different trees then this new tree will be added to the forest by combining the other two. Two main differences need to be introduced to Kruskal’s algorithm in order to calculate straight paths: a first variation to produce straight paths using L(G) is to include the condition that two trees in L(G) are joined only as long as their corresponding vertices and edges in G still form a path as it has been defined above (a description of how to do this is given in the next paragraph). Secondly, in order to produce “straight” paths in G, we process only the edges in L(G) with an “angle” below a preset value $\alpha$. The resulting forest F will consist then of all the paths in G following the constraints given above for the straight path graph SPG.

While the use of L(G) in the explanation is useful, its explicit representation is not necessary. Instead, the pairwise relations of edges incident to the same vertex in G (the equivalent of the edges in L(G)) can be directly process sequentially, from the one with the lowest “angle” or cost to the one with the highest. In the pseudocode implementation of the algorithm given below a type of object encodes these pairwise relations between incident edges and can be sorted by its associated cost. A necessary condition to join trees in the forest (paths in this case) is that they remain paths; any branching needs to be excluded. This means that every
union of paths through a vertex v of G will invalidate any other potential union involving any of the two edges incident in v. It is convenient then to be able to access the pairwise relations of edges that have been generated by v and which have as one of their members the two edges involved in the union (using a map, for example), and remove them from the sequence of pairwise relations to be processed. The other basic condition from Kruskal to add edges, that is, that they belong to separate trees, is still valid. This will prevent the formation of cycles, in case of a large α, for example.

**CALCULATE_STRAIGHT_PATH_GRAPH (G, α)**

\[
Q = \emptyset \quad //\text{min priority queue.}  
F = \emptyset \quad //\text{forest as disjoint set}  
\]

**for** each edge \( e \in G.E \)  
MAKE-SET(F, e)  

**for** each vertex \( v \in G.V \)  
**for** each pair of edges \( e_0 \) and \( e_1 \) incident to \( v \)  
allocate new relation \( lge \) \quad //\text{Equivalent to an edge in } L(G).  
lge.cost = \text{COST}(e_0, e_1)  
lge.v = v  
lge.e0 = e_0  
lge.e1 = e_1  
INSERT(Q, lge)  \quad //\text{Orderer according to cost,}  
\text{associate } lge \text{ to } v  

**while** NOT-EMPTY(Q) and MINIMUM(Q) < α  
lge = EXTRACT-MIN(Q)  
if FIND-SET(F,lge.e0) ≠ FIND-SET(F,lge.e1) \quad //\text{Avoid potential cycles,}  
UNION(F, lge.e0, lge.e1) \quad //\text{unite the two sets that contain } lge.e0 \text{ and } lge.e1  
**for** each event \( olge \) associated with \( lge \).vertex  
if \( lge \neq olge \) and SHARE-EDGE(olge, lge)  
REMOVE (Q,olge)  

**return** F  

Once the forest of edges F has been created a second function would process the original G graph and build the straight path graph SPG according to the paths in F. Since the type of representation of this SPG may be dependant of its intended use, the implementation is left open.

This straight forward algorithm, implemented using the C++ programming language, has produced good enough results in the tested circulation data, but it has however a number of potential limitations and problems. Like many greedy algorithms it optimises costs locally, which can lead to “straight” paths made up of segments with small angle values between them, which when added, do not define a very straight path. One possibility to mitigate this potential problem is to deal with it at the cost
calculating function. Otherwise an alternative would be to consider the cost as a function not of the pair of edges involved, but of the paths in \( F \) the edges belong to. Initially these would be the single edges, but every time a union would be performed, costs would need to be updated for all pairwise relations involving edges that belong to the newly joined paths (these will be those at the beginning and end of the newly formed path) and their place in the order of processing modified according to their new cost. This would allow to accumulate deviations from straightness as the paths are built.

Figure 2: A few examples of straight path graphs.
6. RESULTS AND APPLICATION

The resulting straight path graphs, of which its vertices correspond to more or less straight paths in the original circulation graph, and its edges consist of the crossings and adjacencies of these paths, resemble in principle (though not in meaning) the syntax of axial maps, as these have also straight lines as their vertices and crossings as their edges. While there are important conceptual differences between them, both describe succinctly different configurations, in the case of the straight path graphs those of pedestrian, bicycle or vehicular circulation networks. Also, while we have made the effort to deemphasise any idealist conception of these paths and the resulting graph, their development is not simply driven by the economy of their representations, but had its motivation in the potential of the axial map as a diagram. To take segments of circulation networks implying very small changes of direction as the entities of a syntactic description, seem to define a reasonable heuristic for generating descriptions of their configurations. Other features such as cycles, branchings or paths that are instead as curved as possible (by simply inverting the order in which “angles” are sorted and processed) could also provide a catalogue of relevant characteristics to compare configurations.

The straight path graph resulting from this algorithm can be used directly for the calculation of the graph edit distance; this would be then based simply on the topology of this SPG graph, that is, how each vertex (each straight path) is linked to other vertices (how it crosses and connects to other straight paths). The algorithm for the graph edit distance proposed by Riesen and Bunke (Riesen, Neuhaus et al. 2007) can also take graphs in which its vertices and edges have labels with different information, which can be used to calculate the edit distance. Vertices could include the length of each straight path, for example, or edges the angles of incidence of paths. We have tested a number of different such labellings in calculating edit distances between straight path graphs. As the basis for these labellings, we determined that the relative centrality of the vertices (the straight paths) would be a good description of its characteristics within a circulation network. Accordingly, two standard centrality measures in spatial analysis have been used to compare the location of vertices within the graph: betweenness centrality and closeness centrality. Of these two, closeness centrality resulted in better results, that is, graphs showing features that could be meaningfully interpreted as being more similar to each other. One may speculate on a plausible explanation for this: perhaps betweenness centrality shows more the effect of the rest of the graph on each vertex, while closeness centrality emphasises the relative location of each vertex in the graph. This is an interesting conclusion, as it highlights the relevance of early measures developed within the field of Space Syntax for describing the actual configurations, rather than their effects on circulation and other forms of occupation. In the
case at hand the graphs used have little relevance in an analysis of circulations, as these are in most cases too small, and the effect of at boundaries of the graphs too large to provide any meaningful understanding of these measures in relation to occupation. Still, closeness centrality produces a differentiation of the vertices (the straight paths) depending on their location on the graph that characterises them with a useful precision. In “the Social Logic of Space” Hillier and Hanson proposed a number of related measures (relative asymmetry, integration or control, for example) to describe the role of vertices in the axial map and the y or convex map of settlements, urban space and buildings, and to link the configurations of these diagrams to their function as social and cultural interfaces (Hillier and Hanson 1984). This apparent confirmation of the capacity of these measures to describe configurational features in graphs could be an important contribution of Space Syntax to other forms of research in pattern recognition. A further refinement employed was the use of vertex labels that, rather than using closeness centrality directly, multiplied it by the total length of the straight path. This resulted in short paths having a lower substitution cost than longer paths, so longer paths become more relevant than short ones. It is possible that different combinations of configurational and morphological measures into functions for labelling costs of the edit of vertices may also provide interesting results in graph comparison and classification.

Figure 3: Dendogram showing the hierarchical clustering of straight path graphs, using the edit distance.
7. CONCLUSION

We have explained a development of a diagram loosely related to the axial map, and its use in producing representations of circulation configurations that can be automatically compared and classified using pattern recognition algorithms. The paper has, besides describing the details of this diagram, also tried to contextualise its origins and use. This contextualisation has been based on a critique of the excessive weight placed on the instrumentality of diagrams to producing metrics, instead of looking at the qualities of their representation in terms of the type of spatial entities and their syntaxes. While recuperating some of the original intentions of the “morphic languages” that were the basis of the research program of Space Syntax, the paper has also discussed the practical limitation of understanding these as ideal objects, and proposed instead an operational explanation of these diagrams that accounts for the contingencies of their genesis and considers them as instruments of conceptualisation, rather than as repositories of essential properties of space.

We should reinstate in this conclusion that it is not so much the specific diagram of the straight path graph that should be the focus of this paper, but more the type of diagrammatic entities proposed. The use of pseudocode makes clear that this is an operation for producing a type of diagram, and not a description of ideal qualities of space: the fact that the algorithm can be improved and modified emphasises its operational, rather than idealist, intention. The straight path graph explained here is then not so much an idea in search of an algorithm as an algorithm which can signify different concepts.

The insistence on the syntax of diagrams and their operational contingency, rather than on the measures they afford, is not simply the result of an allegiance to a particular critical or post-structuralist discourse, but also of a pragmatic approach in trying to produce spatial descriptions suitable for computational processing. A similar technical constrain is behind the non-essentialist forms of classification presented in the paper (using the unsupervised machine learning method of hierarchical clustering). It is the conceptualisations afforded by the computer, the type of diagrams it also imposes and the way these organise the objects of its representations, that are responsible for promoting such an outlook. The theoretical implications of the technicity of computers is obviously beyond the space available to develop in this paper as well as its scope and context, but a critical reflexion of the types of ideologies promoted is always necessary if we want to responsibly propose how to use the computer to describe, analyse, and affect the world, in this case through the representation of spatial organisations.

REFERENCES


Proceedings of the 12th Space Syntax Symposium


Dalton, N. S., J. Peponis and R. Conroy-Dalton (2003). To tame a TIGER one has to know its nature: extending weighted angular integration analysis to the description of GIS road-centerline data for large scale urban analysis.


