Efficient Algorithms for Qualitative Reasoning about Imprecise Space

Thodoros Topaloglou*

Department of Computer Science, University of Toronto
Toronto, Ontario, M5S 1A4, Canada

Abstract. This paper addresses the problem of qualitative spatial reasoning and presents efficient algorithms to deal it. We assume a representation which views space as a totality of objects surrounded by a haze area and connected in terms of qualitative spatial relations. Statements relating objects in this represesentation are expressed in terms of haze-order constraints. Reasoning about haze-orders involves, first, determining the consistency of a set of haze-order constraints, and, second, deducing new relations from those that are already known. The developed reasoning algorithms make use of a data structure called haze-order graph which trades space for efficiency. Experimental results illustrate the efficiency of the algorithms.

1 Introduction

Reasoning about the spatial relations between physical objects or regions of space is of fundamental importance. In many cases this can be done without precise quantitative information about these relations. In fact, some knowledge of the topological relations between the entities of interest, combined with incomplete and imprecise information about distances, directions and relative sizes, may be sufficient to draw useful conclusions. Consider the following example:

> An intelligent system is intended to coordinate first-aid vehicles that cover a geographic region. The region is divided into sections, which are further subdivided into subsections. For the coordinator and the vehicle drivers, *landmarks* serve as locations that are known by all concerned. Other spatial information such as vehicle or trouble spot locations are represented relative to landmarks. Each vehicle has a range of activity, called *scope*, which is the area that it can reach within, say, 2 minutes from its current position. Each vehicle reports its position to the coordinating station in imprecise terms (for example,"moving east","at High Park", etc.). Hence, the vehicle position is an *indeterminate* point. The scope of a vehicle is also partly known and is therefore assumed to be a region with an indeterminate boundary. Finally, the location of a trouble spot is reported by the coordinating station in imprecise terms, often through a reference to the nearest street intersection, and is yet another indeterminate point.

* EMAIL: `thodoros@cs.toronto.edu`, WWW: `http://www.cs.toronto.edu/~thodoros`

In the context of this example, the kind of questions for which qualitative spatial reasoning is required, may be: Identify the vehicle that is closest to the trouble spot. Identify the largest city block not in the scope of any vehicle. Which vehicles in the scene are stuck in a traffic congested area? If some vehicles are removed from service how do the rest need to move in order to cover the area better? Many of these questions could be solved by a human operator without consulting precise information. The goal of this research is to develop a representation and a computational model that make the same task possible in a more automated setup.

This paper assumes a representation which is based on the haze-point ontology of space proposed in [Top96], and presents efficient algorithms for qualitative spatial reasoning.

In the haze-point ontology, space is viewed as a totality of spatial objects connected in terms of spatial relations. A haze point is the most elementary object type since higher order objects are composed of haze points. A haze point is a non zero-sized object that is associated with an area of haze such that he point in question may be located anywhere inside it. Haze points are related in terms of haze or precedence relations. The former means that two objects are "close" to each other and therefore indistinguishable. The latter means that the one object (one-dimensionally) precedes the other. Reasoning with spatial objects and their relationships in haze-space essentially amounts to reasoning with one-dimensional haze points and the haze and precedence relations (or the model of haze-orders, for short). Answers to the reasoning problem drawn in this context are then combined to create answers to the reasoning questions using higher-order constructs.

Reasoning about haze-orders involves, first, determining the consistency (satisfiability) of a set of haze-order assertions (also termed constraints), and, second, deducing new relations from those that are already known (i.e., computing the closure of the input haze-order assertions). In this paper we study both reasoning questions in the context of one-dimensional haze-order space.

The rest of this paper is organized as follows. Section 2 presents an overview of the structure of haze space. Section 3 introduces the structure of the qualitative graphs that underlies the reasoning algorithms developed in Section 4. Finally, Section 5 summarizes the contributions of the paper and presents an initial experimental evaluation which illustrates the efficiency of the algorithms.

2 An Overview of Haze Space

Figure 1 shows the simple case of one-dimensional points depicted as small intervals. An interval representing a haze point, denotes the positional likelihood the actual point. Points can be either in *haze* ($h(x, y, g)$) or *precedence* ($x \prec_g y$) relations with each other; g is a parameter denoting the size of the interval and x, y are one dimensional haze points. Relation h is symmetric, reflexive and non-transitive; $\prec_g$ is asymmetric and transitive; in addition, h and $\prec_g$ are complementary, i.e., $\forall xy(x \prec_g y \vee y \prec_g x \vee h(x, y, g))$. A formal presentation of the haze space appears in [Top96].

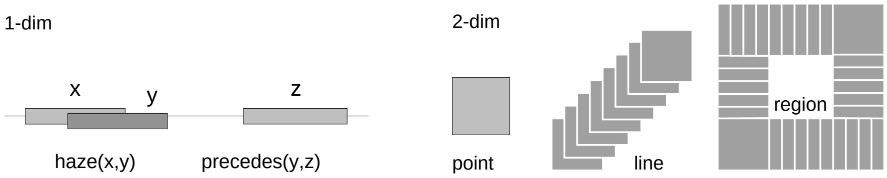

Fig. 1. The *haze*-point space

In the two-dimensional, case points are surrounded by a square-shaped [2] haze. Two points can be combined to form a line, in fact a haze-line. A sequence of connected haze-lines defines a haze, and a closed haze defines a region. The model can be extended to more than two dimensions. The only restrictive assumptions that exist are that the modeling space is orthogonal and discrete, which are nevertheless realistic assumptions for geographic space. The discreteness assumption implies that g, the measure of the haze size, takes integer values. The orthogonality assumption implies that spatial entities are interpreted over the cartesian space.

In [Top94, Top96] we have presented a set of spatial relations on points and regions that are defined in the haze space. We have shown that the first-order theory of space based on the haze and precedence relations, for a constant g, is complete (i.e., has a unique model up to isomorphism). We have also shown that this theory of two-dimensional space is the combination of two independent one-dimensional theories. The practical advantage of this result is that reasoning about two-dimensional relations is reduced to two independent reasoning problems, one for each dimension, and the result can be translated back to the higher dimensional terms.

The definition of spatial relations in terms of the haze and precedence primitive relations is, in essence, equivalent to indexing spatial relations with an explicit argument which stands for the degree of precision. Changes to the degree of precision obviously affect the truth or falsehood of stored assertions as well as that of the implied ones.

3 Haze-Order Graphs

In this section we introduce the definitions and the theorems on which reasoning about haze-orders is based. We start with the definition of the haze-order graphs, that is a graph representation for a set of haze-order constraints.

Definition 1. A haze-order graph is a labeled graph whose vertices represent one-dimensional haze points and its edges represent binary haze-order relations that hold between points. Edges are denoted by triples (x, l, y) and are either directed (l is $<$) standing for $\prec$ relations or undirected (l is g) standing for h relations.

[2] The square-shaped two-dimensional points are justified by the law of *independent combination* [Top96] that is employed to construct thoeries of k-dimensional space out of k one-dimensional coordinate theories.

Figure 2 illustrates several haze-order graphs. For instance, graph (e) represents the set of haze-order constraints $\{x \prec z, h(x,y), h(y,z)\}$.

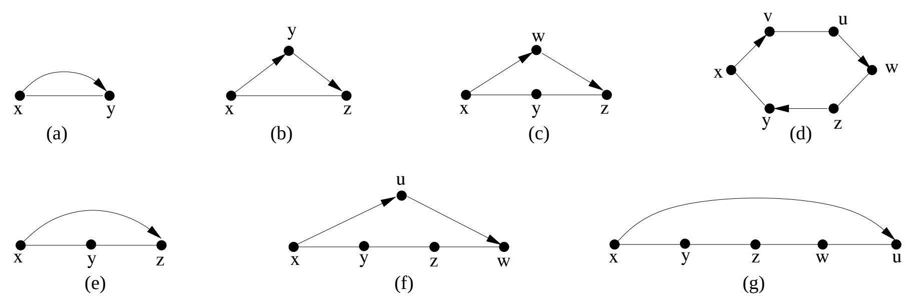

Fig. 2. Inconsistent (a-d) and consistent (e-f) $<^i, g^j$-cycles.

Definition 2. In a haze-order graph hG, a sequence of n successive edges (x_i, l_i, y_i) defines a path of length n. If all the labels along a path are $<$, then the path is a $<$-path. A path $(x_1, l_1, y_1)\ldots(x_i, l_i, y_i)$ where $x_1 = y_i$ is a cyclic path. A cyclic path with both $<$ and g edges is called $<g$-cycle. An $<^i, g^j$-cycle is a $<, g$-cycle with i occurrences of $<$ edges and j occurrences of g edges.

Definition 3. A haze-order graph is consistent if its vertices can be interpreted as elements of a totally ordered set that satisfies all the haze-order relations that exist in the underlying constraint set.

Note that there is more than one ordering that can satisfy a set of haze-order constraints. For example, the constraints in $C = \{x \prec y, h(y,z)\}$ are satisfied by the orderings $v(x) < v(y) < v(z)$ and $v(x) < v(z) < v(y)$, $v(x)$ denotes an interpretation of v. Both orderings are models of C.

A model of a haze-order constraint set is a mapping from vertices to a totally ordered set, say P, such that the values assigned to any two vertices satisfy the relation represented by the edge connecting them. A model also assigns a value g to the parameter that stands for the size of the *haze*

Definition 4. Given a haze-order graph, $hG = (V, E)$, a model is a quintuple $(P, v, \mathsf{g}, R_<, R_g)$ where P is a totally ordered set, v is a mapping from its vertices V to P, g is a constant denoting the haze size, $R_<$ is a binary relation whose elements are pairs of P elements such that for every $(x, <, y) \in E$, $\langle v(x), v(y)\rangle \in R_<$ and R_g is a binary relation whose elements are the pairs of P such that for every $(x, g, y) \in E$, $\langle v(x), v(y)\rangle \in R_g$.

The set of integers, $\mathbb{Z}$, along with the relations $\mathbb{Z}_<$ and $\mathbb{Z}_g$ will be the underlying interpretation structure for the haze-order graphs in this paper. For some g, $\mathbb{Z}_<$ and $\mathbb{Z}_g$ are defined as
$\mathbb{Z}_< = \{(a,b) | a, b \in \mathbb{Z} \text{ and } b - a > \mathsf{g}\}$ and $\mathbb{Z}_g = \{(a,b) | a, b \in \mathbb{Z} \text{ and } |a - b| \leq \mathsf{g}\}$

Theorem 5. *A haze-order graph is consistent if it has at least one model.*

Proof. By definitions 3 and 4.

Theorem 6. *A haze-order graph is consistent if and only if it does not contain any $<^i, g^j$-cycles with $i \geq j$.*

Proof. (**Only if**) If $j = 0$ the cycle is inconsistent due to the irreflexivity of $<$. If $j \neq 0$ and $i > j$, then we can cancel any g-edge on the path and its preceding $<$-edge and still obtain a $<$-cycle which induces inconsistency. If $i = j$ an extreme case is encountered in which all the g-edges, (x, g, y), need to be interpreted by an ordering where y precedes x (i.e., the g-edges materialize to the counter direction of the $<$-edges), and all the $<$-edges, $(z, <, x)$, are tight (i.e., the distance between z and x is just above the g threshold). As a result, for any pair of successive $<$ and g-edges, e.g., $(z, <, x) \wedge (x, g, y)$, z will precede y. Applying this inductive argument for the $<^i, g^i$-cycle starting from a vertex u, we will eventually find that u precedes itself, since the cycle closes at u, leading to a contradiction and therefore to inconsistency. The $i \geq j$ condition is illustrated in Figure 2. Graphs (a)-(d) satisfy the condition and therefore are inconsistent whereas graphs (e)-(g) are consistent. In particular, graph (f) is a $<^2, g^3$-cycle with a $\langle v(x), v(y), v(v), v(z), v(w)\rangle$ being a consistent model for it.
(**If**) We need to show that any consistent graph does not contain a $<^i, g^j$-cycle with $i \geq j$. We use inductive on the number of edges. For the cases of 1 (trivial graph), 2 and 3 vertices, the base cases, this is verified by inspection of graphs (a), (b) and (e) in Figure 2. In the induction step, we assume a consistent graph without an $<^i, g^j$-cycle, $i \geq j$, and insert an edge. Adding an $<$-edge will leave the graph consistent as long as it doesn't close a $<$-cycle or be the ith $<$-edge in a $<^i, g^j$-cycle, $i = j$. In any other case a consistent model exists. The constraint imposed by the new edge refines the allowable existing orderings of the graph vertices by eliminating those that violate it. Adding a g-edge, as long as it is not the ith g-edge that closes a $<^i, g^i$-cycle, leaves the consistency of the graph unchanged. □

From the discussion in the proof of Theorem 6 we conjecture additional conditions that a consistent haze-order graph with $<^i, g^j$-cycles must satisfy. Observe the graph in Figure 3(a). It contains a $<^2, g^3$-cycle, and thus it is consistent. The ordering $\langle v(x), v(w), v(y), v(v), v(u), v(s),\ v(z), v(t)\rangle$ is a potential model for it. The condition that must be satisfied in order for the above ordering to become a model is that all the order and the haze relations are satisfied. This requires that there exists an assignment from vertices to integer values and an integer value g that satisfies all the constraints. The selection of allowable g values involves resolving some additional metric constraints. The expression $i*(\mathsf{g}+1)-j*\mathsf{g} = 0$ determines the lower bound for the g value. The meaning of the expression is that in the boundary case, that is, $i = j - 1$, all the g-edges are ordered in the counter direction of the $<$-edges, and $<$-edges are all tight (i.e., just above the g threshold). In this case a g exists such that it guarantees that the length cycle remains zero and therefore is consistent. If $i < j - 1$ (non-boundary case), this condition changes to $\sum_{n=1}^{i}(\mathsf{g} + a_n) - (j - k) * \mathsf{g} + b_k = 0$ where a_i is an integer increment (≥ 1) by which each $<$-constraint exceeds the g threshold, $k > j - i$ is the number of g-edges, (x, g, y), that are interpreted by a $\langle v(x), v(y)\rangle$ ordering, and $b_k = \sum_{l=1}^{k}[v(y) - v(x)]_l$). The cycle in Figure 3(a) requires that g is at least 2 due to the above condition. Hence, a model can be $\langle \mathsf{g} = 2, v(x) = 0, v(w) = 2, v(y) = 3, v(v) = 4, v(u) = 5, v(s) = 5, v(z) = 6, v(t) = 8\rangle$. Figure 3(b) shows the sorted cycle.

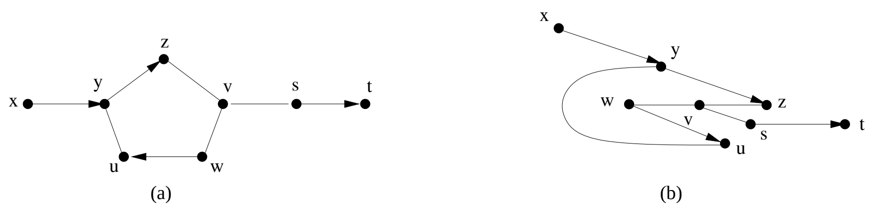

Fig. 3. Ordering $<^i, g^j$-cycles.

The minimum g value for each $<^i, g^j$-cycle determined by the above expression, defines the maximum degree of detail for which a consistent assignment for an $<^i, g^j$-cycle exists. Therefore, the maximum g value that makes all $<^i, g^j$-cycles of a haze-order graph consistent will characterize the minimal model among the many possible models of the graph.

Definition 7. Two haze-order graphs are logically equivalent if they have the same set of models.

Definition 8. Given a haze-order graph, a (x, g, y) edge is disambiguated to the left (resp. right) if the ordering $\langle v(y), v(x)\rangle$ (resp. $\langle v(x), v(y)\rangle$) cannot occur in any of its graph's models.

Notation: Label $\lhd$ denotes a "close-order" relationship; $(x, \lhd, y)$ says that although x, y are close to each other, x is slightly preceding y. Disambiguation of g-edges introduces a new edge label and therefore a new type of edges. $\lhd$-paths ands $<, \lhd, g$-paths are also defined in a similar fashion as the $<, g$-paths.

Proposition 9. *In a haze-order graph, a g-edge between vertices x, y is disambiguated,*

1. (g-edges over triangles) *if vertices z,x and y are connected by a g^2-path and z, y by an $<$-edge, then the g-edges disambiguate to $(z, \lhd, x)$ and $(x, \lhd, y)$* (Figure 4)*;*
2. ($g^i, <^{i-1}$-cycles) *the i in the number g-edges of a $g^i, <^{i-1}$-cycle are disambiguated in the counter direction of the $<$-edges* (Figure 3).

Long g-paths represent many possible orderings. The consequence of this proposition is that in certain cases we can prune some impossible orderings by just using the structure of the haze-order graph. This is beneficial in the design of an order inferencing algorithm.

Proposition 10. *The vertices of any g-cycle with $2n+1$ edges can be interpreted inside an interval with maximum length $n\mathsf{g}$.*

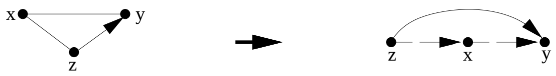

Fig. 4. Disambiguating g-edges

Definition 11. A *disambiguated* haze-order graph is a haze order graph with all the g-edges possible to disambiguate, disambiguated.

The corresponding disambiguated graph, dG, of some haze-order graph, hG, has the same set of vertices and edges as hG and a richer set of labels.

In the following, we wish to entail the relations that hold between any two haze points x and y given a haze-order graph representation. Most importantly, we want to entail the strongest relations between any two points. For any two points, we want to decide whether the one precedes the other (i.e., $x \prec y$) or are in haze relation (i.e., $h(x,y)$) or if either relation can hold (i.e., $x \prec y \vee h(x,y)$) or, finally, if the universal relation holds (i.e, $x \prec y \vee h(x,y) \vee y \prec x$). The meaning of the universal relation is that any relative positioning of the two points is possible.

Notation: If the expression $r_1(x,y) \vee ... \vee r_n(x,y)$ is entailed by a haze-order graph, then we say that relation $R = \{r_1,..,r_n\}$ holds between the two points x, y.

The two lemmata below present a graph-theoretic definition for the notion of entailment.

Lemma 12. *Let hG be a consistent haze-order graph and x, y are two vertices of hG; hG entails*

1. *$x \prec y$, if there is an $<$-edge or an $<$-path or a $<^i, g^j$-path with $i > j$ between x and y;*
2. *$x \prec y \vee h(x,y)$, if there is a $<^i, g^j$-path with $j = i \vee j = i-1$, between x and y;*
3. *$h(x,y)$, if there is g-edge between x and y;*
4. *the universal relation in all other cases.*

Proof. Since the haze-order graph is consistent, there are no forbidden $<^i, g^j$-cycles. However, there might be multiple paths connecting any two of its nodes. First, we consider single paths. The entailment of $\prec$ relationships is justified by the transitivity of $\prec$ if the two nodes are connected either by a $<$-edge or an $<$-path, and by the quantitative constraint stated in the proof of Theorem 1. If there is a $<^i, g^j$-path between x and y, i.e., if all the g-edges disambiguate to the opposite direction of the $<$-edges, then still due to the the definition of $\mathbb{Z}_<$ and $\mathbb{Z}_g$ and even if the $<$-edges are defined just above the g threshold, the $\prec$ relation is the only possibility between x and y. The $\{h, \prec\}$ relation is derived if the $\prec$ relation can not win over a $<^i, g^j$-path. This happens if the number of g-edges is equal to or exceeds by one the number of $<$ edges as suggested by the earlier quantitative formula. $\{h, \prec\}$ cannot be derived if $j > i+1$ since the $<^1, g^3$-path, $\{x \prec y, h(y,z), h(z,t), h(t,s)\}$, can have a model which satisfies

$s \prec x$. Thus, $<^i, g^j$-paths with $j > i+1$ should entail the $\{\prec, h \succ\}$ relation. The entailment of h is a trivial case. If two nodes are connected by multiple paths then the entailed relation will be the intersection of the relations entailed for each path. □

Lemma 13. *Let dG be the disambiguated graph of a haze-order graph hG and x, y be two vertices of dG; dG entails*

1. $x \prec y$, *if there is an* $<$*-edge or a* $<$*-path or a* $<, \lhd$*-path or a* $<^i, \lhd^k, g^j$*-path with* $i+k > j, k \geq 1$, *between x and y;*
2. $x \prec y \vee h(x,y)$, *if there is a* $<^i, \lhd^k, g^j$*-path with* $i+k = j \vee i+k-1 = j$ *and* $i \neq 0 \vee k \neq 0$, *between x and y;*
3. $h(x,y)$, *if there is g-edge or a* $\lhd$*-edge between x and y;*
4. *the universal relation in all other cases.*

Proof. The proof of Lemma 13 is an extension of the proof of Lemma 12. In particular, $\lhd$-edges appear as result of disambiguating g-edges. These edges allow us to refine some of the $x \prec y \vee h(x,y)$ inferences to $x \prec y$ inferences since they provide evidence that certain haze relations are one-sided. Edges labeled by $\lhd$ entail $h(x,y)$ relations because of Definition 8. □

Proposition 14. *Let hG be a haze-order graph and dG its corresponding disambiguated haze-order graph. If dG entails* xRy *and hG entails* $xR'y$ *then R implies* R' *(but not necessarily the converse).*

Proof. By applying Lemmata 12 and 13.

In the rest of this section, we show that the entailment notion based on haze-order graphs computes stronger relations between haze-points, than the relations computed by the path consistency algorithm.

Qualitative constraint networks are common tools in the study of constraint satisfaction and the entailment of strongest relations in the context of temporal and spatial reasoning. A qualitative constraint network for haze-order constraints is a haze-order graph with a complete set of edges, i.e., each vertex is connected to all other vertices. The intersection and composition operations are key concepts for the consistency theory developed in the context of qualitative constraint networks.

Definition 15. (*Intersection*) If $R1$ and $R2$ are two relations holding between a pair of points x, y, then the combined relation between x,y is defined by their intersection, $R = R1 \cap R2$.

Definition 16. (*Composition*) The composition of two relations $R1$ and $R2$ holding between two pairs of points x, y and y, z, respectively, is defined as $R = R1 \otimes R2 = \cup_{r_1 \in R1, r_2 \in R2} T(r_1, r_2)$, where T the composition operation between atomic relations.

Figure 5 shows the transitivity tables that define the composition operations in the relation set represented in a haze-order graph (T_1) and a disambiguated haze-order graph (T_2). Notice that table T_2 is more detailed around the *haze* relation (labels $\lhd, g, \rhd$).

T_1	$<$	g	$>$
$<$	$<$	$\{<, g\}$	$\{<, g, >\}$
g	$\{<, g\}$	$\{<, g, >\}$	$\{>, g\}$
$>$	$\{<, g, >\}$	$\{>, g\}$	$>$

T_2	$<$	$\lhd$	g	$\rhd$	$>$
$<$	$<$	$<$	$\{<, \lhd\}$	$\{<, \lhd\}$	ALL
$\lhd$	$<$	$\lhd$	$\{g, \lhd\}$	g	$\{\rhd, >\}$
g	$\{<, \lhd\}$	$\{<, \lhd\}$	ALL	$\{g, \rhd\}$	$\{\rhd, >\}$
$\rhd$	$\{<, \lhd\}$	g	$\{\rhd, >\}$	$\rhd$	$>$
$>$	ALL	$\{\rhd, >\}$	$\{\rhd, >\}$	$>$	$>$

Fig. 5. Transitivity Tables

The following definitions are "classics" in constraint networks references [Mac77], [Fre82], [vB92], [GS95], etc.

A constraint network is *arc-consistent* if for each pair of vertices the entailed relation is not empty, and it is *path-consistent* if for any triple of vertices, x, y and z, the condition $R3 \subseteq R1 \otimes R2$ holds, where $R1, R2, R3$ are the relationships between (x, y), (y, z) and (x, z), respectively. Path consistency can be checked in $O(n^3)$ time, where n is the number of vertices in the constraint graph, using the Mackworth's path consistency (PC) algorithm [Mac77].

A constraint network is *minimal* if the relations holding between each pairs of its vertices are the strongest possible. Relation $R1$ is stronger than $R2$ if $R1$ implies $R2$ but not the reverse. An equivalent definition for a constraint network to be minimal is if every subnetwork relative to the overall network is strongly consistent (the relations between its vertices are minimal). The size of the subnetwork depends on the deployed constraint language. [3]

We now investigate conditions that determine minimality in haze-order graphs. A haze-order graph is minimal if the relation entailed for each pair of its vertices is the strongest possible. Unfortunately path consistency can not guarantee minimality in haze-order graphs. Figure 6 shows counter examples. Path-consistency using the transitivity table T_1, will determine that relation $\{<, g\}$ holds between vertices x, y in the first graph, where, by Lemma 12, the relation $<$ is entailed. Path-consistency using the transitivity table T_2, however, yields the correct result for the first graph but it fails to ensure minimality for a graph with a g^3-path such as the the lower graph in Figure 6. Path consistency with T_2 guarantees 5-consistency (any sub-network with 5 vertices is minimal) but not 6-consistency for this graph. An interesting observation that is revealed by this example is that the consistency condition in the presence of a haze relations, depends on the length of the g-paths in the constraint set. Another point is that by enriching the relation set with relations refining the qualitative scope of the original haze relation such as $\lhd$ and $\rhd$, we can increase the degree of consistency that the PC algorithm achieves.

Proposition 17. *The path consistency property can not guarantee strongest relations in a haze-order graph.*

[3] For instance, in the (exact) point algebra (PA) the condition of minimality is that each 4-node subgraph has to be consistent. In the PA without $\neq$ relations the size of the consistent subgraph is 3, etc. [vB92].

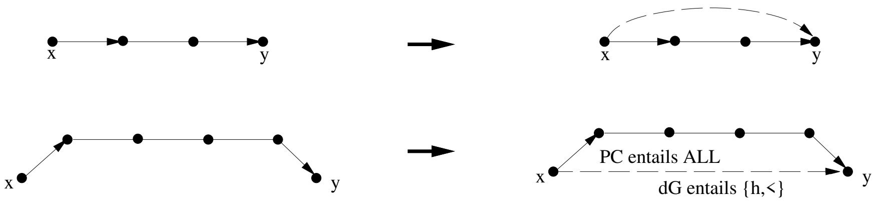

Fig. 6. Minimal Graphs

Proof. By the counter example of Figure 6. □

Theorem 18. *Any consistent haze order graph entails minimal relations.*

Proof. We follow the same argument line as in [GS95]. A minimal relation is the strongest possible relation entailed. We need to show that any stronger relation than the entailed by a haze-order graph is not feasible. We take the case of the haze-order graph with labels $\{<, g, >\}$, for which Lemma 12 establishes entailment. This entailment is strong. The lattice $\{\text{ALL} \rightarrow \{h, <\}, \{h, <\} \rightarrow h, \{h, <\} \rightarrow <\}$ represents all the "weaker to stronger" relation pairs. Let r be the relation on vertices x, y, as entailed by this lemma. Any solution for x, y that replaces r with a stronger relation, r', according to the above lattice will not be logically equivalent, i.e., will not satisfy all the models that a solution containing r satisfies. This is supported by a case analysis of the transitions and the checking of the quantitative expression that corresponds to each entailment as described in Lemma 12. Similar argument applies to labels $\{<, \lhd, g, \rhd, >\}$ and Lemma 13. □

The implication of Theorem 18 is that the minimal network representation for a haze-order graph can be constructed by simply computing the entailed relation for each pair of nodes.

4 Algorithms

4.1 Consistency

Theorem 6 suggests that a haze-order graph is consistent if and only if it does not contain any $<^i, g^j$-cycles with $i \geq j$. We now present an algorithm (referred as **HOC** from hereafter) which enforces this consistency detection criterion. The main idea is to reduce the haze-order graph to a graph that contains $<$-edges only and whose nodes are collections of nodes related by g-edges (from hereafter we will refer to these nodes as *hyper-nodes*). Then, in a second phase, we detect cycles that satisfy the inconsistency criterion.

The algorithm operates on an adjacency list representation of the haze-order graph. Let $Alist(v)$ be the adjacency list that corresponds to node v. For every (v, g, u) edge, u appears in $Alist(v)$ and v appears in $Alist(u)$. For every $(v, <, u)$ edge, u appears in $Alist(v)$ but not the reverse.

A hyper-node is a node created as a result of collapsing nodes connected by g-edges. Each hyper-node has a single "entry" point (node) and possibly many "exit" points. Each enclosed node, v, in hyper-node, h, has a value $d(v)$ denoting the distance of v from the entry point of h according to the breadth-first traversal. If v is an exit point then $d(v)$ is assigned as a label to the hyper-edge emanating from v. The set of nodes enclosed by a hyper-node h is denoted by $nodes(h)$.

The following notation is used in the consistency algorithm: $status(v)$ denotes the status of node v during the course of the consistency algorithm. Initially all nodes are marked as `closed`. If a node is reached by the algorithm, the it is marked `open`, and when its processing is completed it is marked `done`. Function $out\_point(v)$ returns true if there exists an outgoing $<$-edge from node v, otherwise returns false. $HyQueue$ is a queue of potential hyper-nodes. $H\_edgeQueue$ is a queue for gathering boundary h-edges and $<$-edges that are to be be inserted in the graph. $queue(h)$ is a queue which stores the nodes that potentially become part of hyper-node h.

We can now outline the steps of the algorithm. A detailed description and an implementation appears in [Top96].

Step 1 (reduction) : This step builds a graph. It starts from some initial node, v_0, and creates a hyper-node h with $queue(h) = Alist(v_0)$. Queued nodes are inserted in h if they are not out_points and connected to an enclosed node with a g-edge. out_points are queued in $HyQueue$ since they are potential nuclei for subsequent hyper-nodes. Edges that lead to out_points are also queued for further processing. In particular, queued $<$-edges become edges of the new graph. Queued g-edges are used to correct the d values. This step continues until all nodes of the haze-order graph are marked `done`, i.e., placed in some hyper-node.

Step 2 (detection) : Let R be the reduced graph of hG. R is a directed graph. We run the strongly connected components (SCC) algorithm on R to detect cycles. For each cycle found, we test the condition $\#edges \geq \Sigma_i^{\#edges} weight(edge_i)$. If it holds then graph hG is inconsistent.

Theorem 19. *The* **HOC** *algorithm correctly detects inconsistency for any haze-order graph in $O(n + e)$ time.*

4.2 Computing the "Tight" Graph

Given a consistent haze-order graph, the next step is to compute the tight graph, i.e., a haze-order graph in which with the most tight order relations [4]. The tight graph, T, underlying a haze-order graph, hG, has the following properties:

- all the disambiguations that are possible in hG, are made explicit in T;
- any order relation implied by hG, is derivable in T by a path traversal.

A key concern in the development of the algorithm for computing a tight-order graph, is to resolve ambiguities pertaining to $<^i, g^j$-cycles. An interesting situation arises when the $<^i, g^j$-cycles overlap.

[4] For each vertex the immediate next and previous vertices are identified

The *tightening* algorithm operates on the directed graph R constructed during the reduction step of the consistency algorithm. A topological sort of R gives an initial ordering of the nodes which is then further refined. The refinement step uses information about each node's connectivity. From the connectivity of the frontier nodes to the outside ones we can deduce order information about them. For instance, if two nodes v and u of the same hyper-node h_1 are connected to the same node w of an adjacent hyper-node h_2, and it is $(v, <, w)$ and (u, g, w) then v precedes u; if it happens to be (v, g, u) this haze relationship disambiguates to a close-precedes ($\lhd$) relationship.

The refinement phase of the algorithm traverses the nodes of the haze-order graph twice. First, it walks the nodes starting from the nodes of the topologically last hyper-node going backwards. For each node, it determines the set of its direct predecessors. All the g-neighbours of the predecessor nodes are examined and updated. In effect, all the implicit triangles of nodes involving at least one explicit g-edge are examined and tighter predecessors are possibly recovered. A similar traversal is done starting from the nodes of the topologically first hyper-node and going forward. This time tighter successors are sought. At the end of this process we have information about the closest next and closest previous node for each each node v, as well as information about the possible next and possible previous node of v. A detailed description of the "tightening" algorithm is presented in [Top96]. The next step is to create a representation of the tight-order graph which is suitable for query processing.

4.3 Computing the Query Graph

A query graph is used for answering queries. The query graph, Q, that corresponds to a tight-order graph, T, is a logically equivalent representation to T that supports constant-time retrievals. The query graph is computed using a ranking process similar to that of [GA89] and [GS93] which assigns a rank to each node, thus making query processing a lookup-and-compare operation.

In a haze-order graph the assignment of a single ranking value to each node will not produce the desirable result. For instance, due to the haze relations, there are cases where two points can be either before or after each other. The assignment of a single ranking value will not represent this relationship. that some haze relationships have still remained. To handle such cases, the ranking process is designed to assign a low and a high value to each node.

The query graph computation algorithm takes as input a tight-order graph. The algorithm trades space for time. First, it transforms the edge-based representation of the graph to an adjacency list representation. The cost of the transformation step is $Q(n + e)$. In its main step, the algorithm assigns ranking information to the nodes. This step involves a depth first traversal operation starting from a special node called *start*. During the traversal each node is assigned a rank (the value of an incrementing counter) any time it is traversed. In effect, a node may have more than one ranks. The minimum and maximum rank of the node define its index range. The value(s) assigned as rank of a node, depend(s) on the predecessor node rank and the type of the traversed edge. This step is also realized in $O(n + e)$ time.

5 Conclusions

We have studied the problem of qualitative spatial reasoning in one-dimensional haze space. Haze-space is a qualitative representation of space where entities are related in terms of an indistinguishability and a precedence relation.

The main contribution of this work is the definition of a computational model for reasoning about haze-orders. We have defined a data structure, called haze-order graph, which is used to represent constraints on haze-space entities. We have addressed two reasoning questions, namely, the consistency of a set of haze-order constraints, and the strongest relation computation between any two entities. We have developed polynomial algorithms for both problems, and in addition we have defined a quantitative index structure that can support constant time retrievals. Another interesting finding of our study is the proof that the classical path consistency algorithm can not be used for computing the strongest relation between any two entities. However, this algorithm can still be used for inconsistency detection.

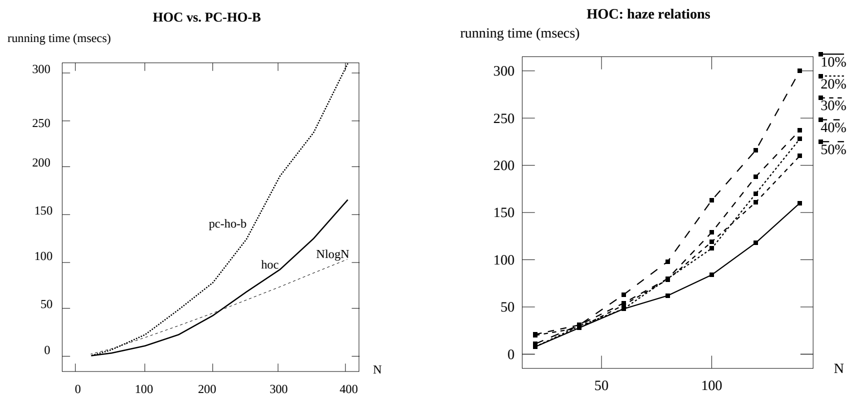

Fig. 7. Performance of Haze-Order Consistency Algorithms

Finally, we have implemented and experimentally evaluated the presented algorithms (a detailed study of the performance of the developed algorithms is presented in [Top96]). The graph on the left in Figure 7, shows a comparison of the path consistency and the haze-order consistency algorithms on randomly generated common data. The graph on the right in Figure 7, summarizes 560 experiments on randomly generated graphs and shows the dependencies of the haze-order consistency algorithm on various characteristics of the datasets (in this experiment we variated the percentage of of $<$-edges in the input dataset).

A number of research issues will be addressed in our future work. Several extensions of the presented algorithms are on our agenda, including extending the algorithms with quantitative information or a variable haze sizes. In addition, we seek to apply the representation and reasoning techniques presented in this paper to other domains that are characterized by partial, “noisy”, and dimensional data.

Acknowledgments. I thank my advisors Professor John Mylopoulos and Professor Janice Glasgow for their insightful comments on the earlier drafts of this paper.

References

[Fre82] E.C. Freuder. A Sufficient Condition for Backtrack-Free Search. *Journal of ACM*, 21(11):958–966, 19982.

[GA89] M. Ghallab and M.A. Alaoui. Managing Efficiently Temporal Relations through Indexed Spanning Trees. In *Proceedings of IJCAI-89*, pages 1297–1303, 1989.

[GS93] A. Gerevini and L. Schubert. Efficient temporal reasoning through timegraphs. In *IJCAI-93*, pages 648–654, 1993.

[GS95] A. Gerevini and L. Schubert. Efficient algorithms for qualitative reasoning about time. *Artificial Intelligence (to appear)*, 1995.

[Mac77] A.K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[Top94] Thodoros Topaloglou. First Order Theories of Approximate Space. In *AAAI-94 Workshop on Spatial and Temporal Reasoning*, pages 47–53, 1994.

[Top96] Thodoros Topaloglou. On the Representation of Partial Spatial Infornation in Knowledge Bases. PhD Thesis, University of Toronto, 1996.

[vB92] Peter van Beek. Reasoning About Qualitative Temporal Information. *Artificial Intelligence*, 58:297–326, 1992.