

A Methodology for Updating Geographic Databases using Map Versions

Ally Peerbocus*, Geneviève Jomier*, Thierry Badard§

Résumé

Cet article traite de la problématique liée à l'échange et l'intégration de données géographiques entre les producteurs et les utilisateurs. Le producteur livre une base de données géographiques à un utilisateur, qui l'utilise comme référentiel pour ses applications spécifiques. Le producteur ainsi que l'utilisateur peuvent mettre à jour ce référentiel. Par conséquent, l'intégration dans la base de données de l'utilisateur de mises à jour que fournit le producteur est une opération complexe à cause des conflits potentiels entre les mises à jour du producteur et celles de l'utilisateur. La base de données résultante risque d'être incohérente et les informations propres à l'utilisateur risquent d'être supprimées inopinément. Les utilisateurs ont donc besoin d'un mécanisme d'aide à l'intégration des mises à jour. Pour cela, nous proposons une méthodologie basée sur l'usage de base de données géographiques multiversion qui permet la détection automatique des différences entre deux versions de carte.

Mot-clefs : Base de données géographiques, mises à jour, version de carte, multiversion

Abstract

This paper deals with issues related to the exchange and integration of geographic data between producers and users. Once a producer has delivered a geographic database to a user, who uses it as a reference for his specific applications, the database may be updated on both sides. Consequently, the integration of updates - delivered by the producer - in the user's geographic database is a complex operation due to possible conflicts between updates performed by both actors. The resulting database may be in an inconsistent state and user's added information may be lost. Therefore, users need mechanisms to

* LAMSADE, Université Paris-Dauphine, {jomier,peerbocus}@lamsade.dauphine.fr

§ COGIT, Institut Géographique National, Thierry.Badard@ign.fr

help them in the process of update integration. This paper provides a methodological framework for the updating of geographic databases. It relies on a multiversion GIS, allowing an automatic detection of conflictual updates between two map versions.

Key words: Geographic database, updates, map version, multiversion

1 Introduction

Geographic Information Systems (GIS) are increasingly used in a large spectrum of applications. Since implementing such systems is complex, users generally, purchase reference geographic data from producers in order to set up their GIS. For instance, a transportation company purchases from a producer a geographic database representing the road network of a given region for its transport planning application. For the user, the database delivered by a producer serves as a *reference map* to develop the application. Geographic data producers are responsible for producing and maintaining up-to-date databases, delivered to users on a regular basis. Meanwhile, users may need to add information on the reference map, or update the map to take into account real world changes, or information they are interested in for instance bus lines and bus stops (see Fig. 1).

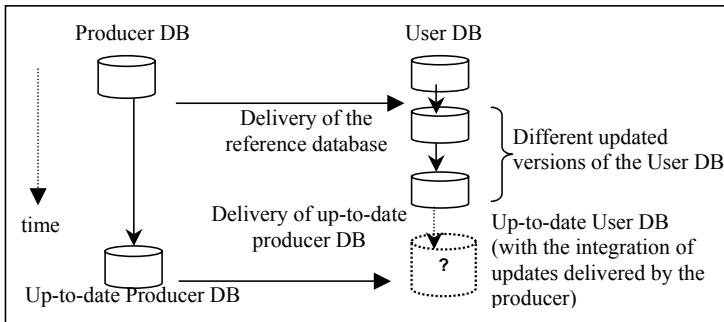


Fig. 1- *Integration of updates*

Consequently, the integration of the producer's updates in the user's database may result in conflicts with those already performed by the user as described in (Badard, 1998a; Badard, 1998b). For instance, if the producer changes the location of a road, to increase information accuracy, the bus line and bus stops along the road must be changed too; otherwise, the user's database is in an inconsistent state.

The first step for a proper integration of these updates requires the identification of the updated objects in the user's database, as well as those in the producer's database in order to detect possible conflicts. At present, producers generally deliver a whole up-

to-date database to the user. Even if the percentage of change is small between two updates, current GIS do not provide any mechanism for the extraction of updates between two versions of a database representing the same area at two different times (Raynal, 1996). Several techniques have been proposed to achieve this purpose. They are based on the exhaustive comparison of all the objects in the two versions of the database (Badard, 1998a), relying on geographic data matching algorithms (Lemarié & Raynal, 1996). Such an approach, well adapted in a general context where no hypothesis on the data model is assumed, is based on complex algorithms and needs a tremendous effort to be implemented.

This paper proposes a mechanism for an automatic detection of conflictual updates performed in two different versions of a database. It is based on the version approach proposed in (Cellary & Jomier, 1990). This paper is organized as follows: section 2 describes the context through an example and presents an overview of related work; section 3 details our approach and the way it is implemented; section 4 concludes the paper.

2 Context and Related Work

Before presenting an overview of related work, this section describes an example of exchange of geographic data between a producer and a user, illustrating conflicts between producer's and user's updates.

2.1 Context

It is based on a road network application relying on Georoute®, a database produced by the IGN (the French National Geographic Institute) and dedicated to car navigation services. Fig. 2 depicts part of the producer's map, identified as $prod_0$, delivered in a geographic database to the user. The map shows the state of the modeled road network, identified as R_1 to R_5 , and land parcels, identified as P_1 to P_6 . Fig. 2 also represents the new up-to-date producer's map, identified as $prod_1$, reflecting the new state of the road network after:

1. the construction of a new road R_6 , splitting parcel P_2 into P_{2a} and P_{2b} ;
2. the deletion of R_2 ;
3. the construction of a new roundabout, identified by P_7 , at the junction of roads R_1 , R_3 , R_4 , R_5 and R_6 , implying the update of all these road sections.

On his side, the user has updated the initial geographic database to obtain a new version of the map, identified as $user_1$, different from $prod_1$. The updates performed in $user_1$ are illustrated in Fig. 2:

1. the update of roads R_1 and R_5 ;
2. the deletion of R_2 ;
3. the creation of a road section R_7 at the limit between P_4 and P_5 ;

4. the creation of an antenna A , representing the user mobile phone company.
 Finally, the map identified as $prod-user_1$ corresponds to what the user would like to obtain after the integration of updates from $prod_1$ and $user_1$.

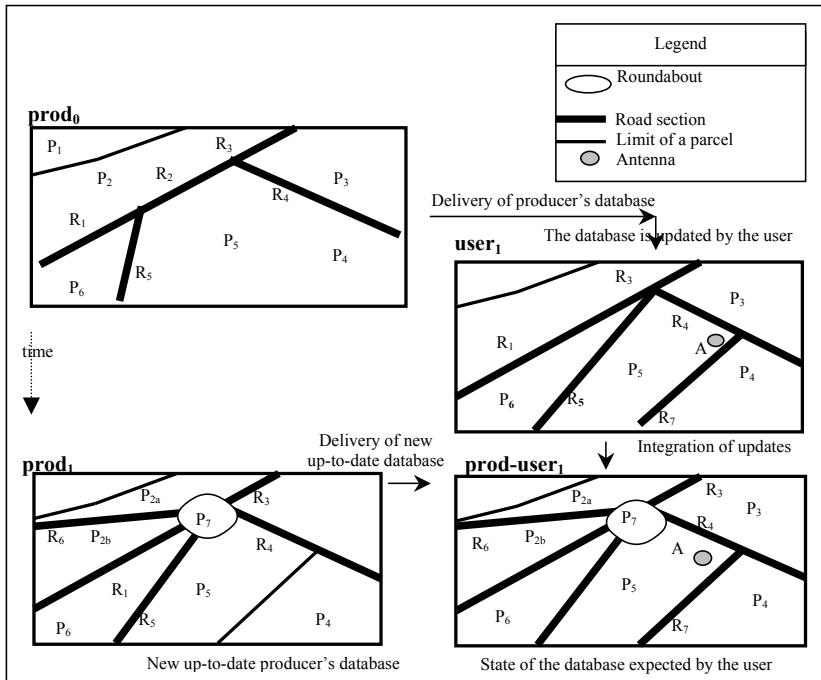


Fig. 2 - The different states of the producer and the user maps

In general, updates in a geographic database can be performed on both the schema and the objects. Updates on an object represent its evolution, i.e., creation, deletion, and thematic and/or geometric changes. Schema updates are required when new kinds of objects appear as bus-stops, mobile phone antennae that are specific objects for the user and whose representation is not provided in the producer's schema of the database. When the producer delivers the new up-to-date database to the user ($prod_1$ in Fig. 2), the latter must consider the new information from this database in order to update his current database ($user_1$ in Fig. 2). He cannot just replace the old reference database with the new one since the resulting database may be in an inconsistent state; for instance, if a road section is enlarged in $prod_1$, he should displace

any antenna on this road. Besides, specific information added by the user may be lost, and some of his updates may be in conflict with the producer's updates, like road R_7 , that exists only for the user.

2.2 Sources of conflicting updates

Several situations may result in conflicts between updates separately performed by the producer and the user. First, conflicts may be due to the update of the same object by both actors, defined as a 1-to-1 update, such as parcels P_5 , P_6 , and roads R_1 , R_5 , updated by the producer and the user. Second, conflicts are very likely to occur in case of group updates like:

- 1-to-N update: one object is deleted and is replaced by several objects, e.g., the splitting of parcel P_2 into P_{2a} and P_{2b} in map $prod_1$,
- N-to-1 update: several objects are deleted and replaced by one object, e.g., the merge of R_1 and R_2 in $prod_0$ resulting in R_1 in $prod_1$,
- M-to-N update: several objects are deleted, and in their place, several other objects are created, e.g., the creation of the new roundabout P_7 by the producer in $prod_1$.

A complete taxonomy of conflicts hindering the updating of geographic database is described in (Badard, 1998a).

2.3 Related work

Several techniques dealing with the detection of update differences between two geographic databases, modeling the same region, have been proposed. They are generally based on the comparison of the geometry (Devogèle, 1998). (Badard & Lemarié, 1999) propose to isolate these differences by using the “geographic data-matching method”, which goes through every database object in a region, and computes the correspondence relationships between objects, from their geometry stored in the two versions. The resulting relationships can be classified, considering their cardinality: a) 1-to-0 or 0-to-1: an object of one database does not match with any object of the other one; b) 1-to-n or n-to-1 with $n>0$: an object of a database matches with one or several objects of the other one; c) n-to-m with $n>1$ and $m>1$: several objects of a database match with several objects of the other one.

The correspondence relationships are then analyzed and updated objects are classified according to the evolution they have undergone (the typology is defined in Badard, 1998b) and this can either concern the object level only or both schema and object levels. Furthermore, new delivery modes dedicated to the exchange of updating information have been proposed (IHO, 1996; Poupart-Lavoie, 1997; Badard 1998b; Badard & Richard, 2001) to help the integration process in databases.

Together with these methods for the detection of updates in geographic databases, propagation mechanisms of these effects have been proposed in a multi-scale database context (Badard, 2000; Kilpeläinen, 1997; Uitermark *et al*, 1998). In all these papers, no hypothesis is made on the data model used and a general solution is provided. Consequently, detecting changes in the whole database requires tremendous efforts and sophisticated algorithms.

A proper updating of a user's database implies the preservation of the integrity of the map delivered by the producer. This means that users must perform their updates on versions of the reference map, and the comparison of the different map versions should be possible in order to detect changes between two map versions. However, as far as we know, in the geographic context only one technical paper of *SmallWorld* GIS (Easterfield, Newell & Theriault, 1992) has focused on the management of version in GIS. But, little implementation details have been provided.

We propose a methodology for the updating of geographic databases called *Updating by Map Versions (UMV)*. It is based on the use of a multiversion geographic database as described in (Bauzer & Jomier, 1993), which allows to manage map versions. The detection of conflictual updates is based on the automatic identification of all the database objects, and not on the comparison performed on the geometry of geographic objects as proposed in (Badard, 2000). The next section deals with the main features of the UMV methodology, and describes how it is implemented.

3 Updating by Map Version Methodology

From now on, this text will respectively refer to the producer database and the user database as *producer-DB* and *user-DB*. Initially, the *producer-DB* contains one version of the map representing the modeled geographic area. This version is identified as *prod₀*. The UMV methodology, comprising of four steps, is illustrated in Fig. 3 and detailed in the next sub-sections.

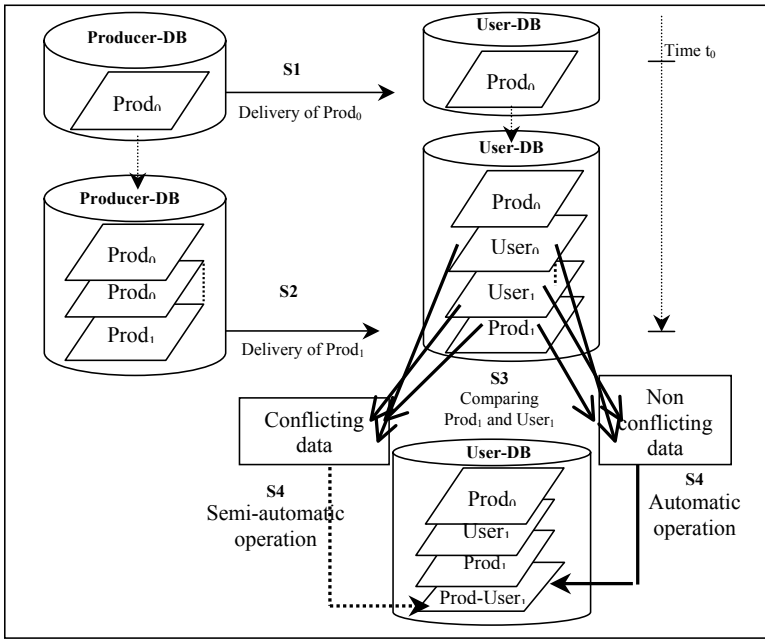


Fig. 3 - The steps comprising the UMV methodology

These steps are:

- S1. At time t_0 , the producer delivers to the user the initial reference map version, $prod_0$. This map version is inserted into $user-DB$, and identified as $prod_0$. It serves as the reference map for the user. $Prod_0$ is preserved in the delivered state, frozen in both $user-DB$ and $producer-DB$. Updates are performed on successive map versions generated from $prod_0$ on both sides. The new generated map versions are identified as $prod_{0,i}$ in $producer-DB$ and $user_{0,j}$ in $user-DB$.
- S2. Then, at time t_1 , an up-to-date database version is delivered by the producer to the user, identified as $prod_1$. The user's map version at this moment is identified as $user_1$. The delivered map version, $prod_1$, is inserted into $user-DB$.
- S3. The map versions $user_1$ and $prod_1$ in $user-DB$ are compared to detect conflicting (from an updating point of view) and non conflicting data.
- S4. Finally, using the strategy, discussed in section 3.3, a new map version is created, $user-DB$, to include part or all of the user and producer's updates. An automatic integration of data can be used for non-conflictual updates. A semi-automatic operation is needed to integrate the conflictual data and the consequences of the conflicts in the final user map version, identified as

prod-user₁. For instance, the bus line of the user present in *user₁* is moved to follow the new location of a road recorded in *prod₁*.

The UMV methodology is supported by a multiversion database (Gançarski & Jomier, 1994), which is introduced in the next sub-section. We assume that both the producer and the user have a multiversion geographic database. The underlying version mechanism is now detailed.

3.1 The multiversion geographic database

Two levels are distinguished: the user level and the database level. At the user level, for external users, the multiversion database appears as a set of independent map versions, representing the same area, which coexist within the same storage space. This means that each map version can be managed (read and updated) separately and independently from the other map versions of the same region. A new map version is always generated or *derived* as a copy of an existing map version.

At the database level, however, one important feature of the multiversion database mechanism is that it automatically allows us keeping track of all database objects that compose a consistent map version. Thus, several versions modeling the same real world object may coexist in the database. This gives origin to the concept of *multiversion object*, which is a “repository” of all versions of a given object, i.e., multiversion object O encapsulates the mapping between all different states of an object, and the corresponding map versions. One of the main advantages of the mechanism is that it minimizes storage occupancy and avoids redundancy while storing multiple map versions. The derivation relationships among the distinct map versions are recorded in a structure called *map version tree*. Updates to objects in one map version are handled without side effects on other map versions, due to an appropriate management of internal version identifiers. To obtain the value of an object in a map version, the system applies a rule stating that it has the same value as that in the map version from which it is derived, except if another value is explicitly specified. This rule is recursive and called *implicit sharing rule*.

When an object is deleted in a map version, its value in the database is set to \perp , meaning that it does not exist. When an object O in a map version v is involved in a group operation - splitting (e.g., the splitting of parcel P_2 into P_{2a} and P_{2b} in map version *prod₁*) or merging operation -, the link between O and the resulting object(s) is stored in a *genealogy graph* (Sperry, Claramunt & Libourel, 1999). A special value “#” for O in map version v is used to denote a group operation. When geographic objects are created from one or several other geographic objects, i.e., the object has one or several ancestors, the resulting geographic objects are initialized with a special value “*” in the map version parent of the map version in which the operation has been performed.

Thus, the genealogy graph represents 1-to-N, N-to-1 and N-to-M evolution of geographic objects.

The system uses the internal identifier of objects to follow their evolution through time. Internal identifiers are managed only by the system, conversely to external identifiers, managed by users. For further details on this approach, the reader is referred to (Cellary & Jomier, 1990; Bauzer-Medeiros & Jomier, 1993; Gañçarski & Jomier, 1994; Cellary & Jomier, 2000).

3.2 Illustration of the multiversion geographic database

The top of Fig. 4 shows a part of the producer multiversion database corresponding to map versions $prod_0$ and $prod_1$ described in Fig. 2. For sake of clarity, we ignore the intermediate map versions between $user_0$ and $user_1$ in $user-DB$, considering that the database is composed only of $user_0$ and $user_1$. Each multiversion object in this figure is represented by a table with two columns: $MV-id$ (for multiversion identifier) and $Value$. Parcel P_6 has a different value for each of the two map versions: $valp6$ in $prod_0$ and $valp6a$ in $prod_1$. The Parcel P_2 in $prod_1$ has been split and replaced by parcels P_{2a} and P_{2b} as illustrated by the genealogy tree. Parcel P_1 has only one value, represented by $valp1$, corresponding to map version $prod_0$. According to the implicit sharing rule and the producer's map version tree, $valp1$ is also the value of P_1 in map version $prod_1$.

The bottom part of Fig. 4 shows a part of the user's multiversion database. Road section R_7 and the antenna A have only one value, $valr7$ and $valA$, respectively, corresponding to map version $user_1$, meaning that they have been created in $user_1$. Parcel P_1 has only one value, $valp_1$, for map version $prod_0$; thus, its value in map version $user_1$ is implicitly shared with that in $prod_0$. Parcel P_6 has two values, $valp6$ in map version $prod_0$ and $valp6x$ in map version $user_1$, because it has been updated in map version $user_1$.

Let us note that the identifier of new objects created in $prod_1$ or $user_1$ must not be in conflict. This can be the case if the same identifier is used in $prod_1$ and $user_1$ to represent two different real world entities. In order to prevent this, the identifier of new objects is prefixed with the name of the database in which it is created. For example, the identifiers of P_{2a} and P_{2b} in Fig. 4 are in fact $prod-DB.P_{2a}$ and $prod-DB.P_{2b}$. For simplicity sake, this does not appear on the figure.

This sub-section has described the main principles of the multiversion approach. In reality, the geometry of some geographic objects may be represented by complex objects. As a consequence, updates are carried out on elementary objects composing the geometry (Peerbocus *et al*, 2001).

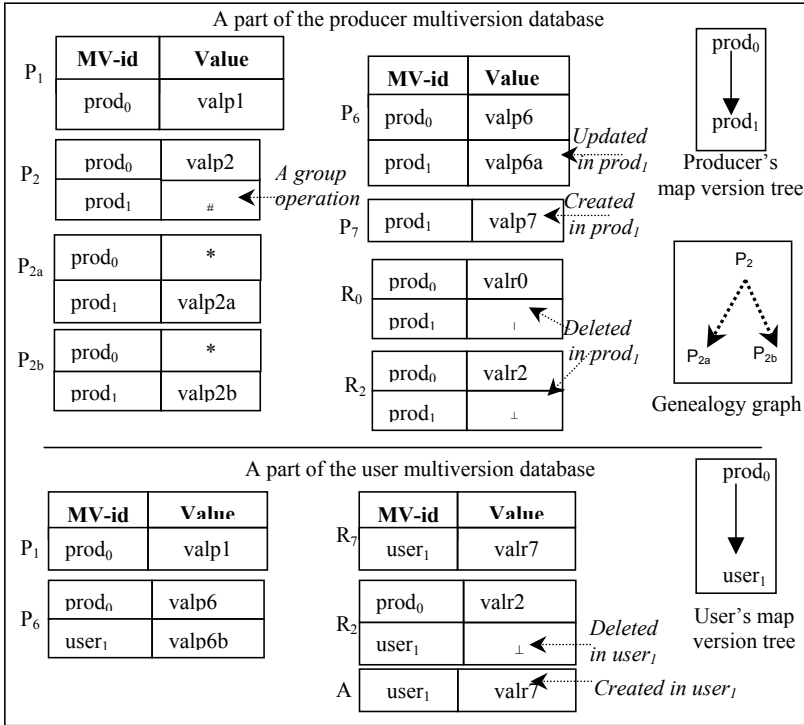


Fig. 4 - Part of the multiversion databases

3.3 Delivery of producer's updates

On delivery, the producer's updated map version *prod₁* is inserted in *user-DB* to enable the detection of conflicts between the producer's and the user's updates. This insertion operation is performed automatically as follows:

1. The system first modifies the map version tree to include map version *prod₁* as derived from *prod₀*; *prod₁* and *user₁* become alternative map versions, both derived from *prod₀*, as illustrated in Fig. 5.
2. The system then verifies, for each object in the delivered map version *prod₁* whether the object has been updated or created by the producer, i.e., it has a value explicitly associated with *prod₁*. If so, the system inserts the value corresponding to *prod₁* in the corresponding multiversion object in *user-DB*.

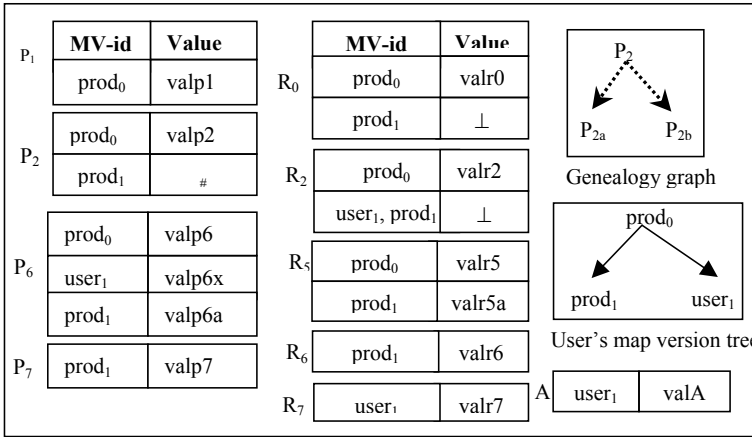


Fig. 5 - Part of the user's database after the insertion of *prod₁*

Finally, the multiversion database is composed of multiversion objects having values corresponding only to *prod₀*, and/or *prod₁* and/or *user₁* (see Fig. 5). For instance, parcel *P₆* has three distinct values corresponding to map versions *prod₀*, *prod₁* and *user₁* respectively; parcel *P₁* has only one value for *prod₀*, shared implicitly by *user₁* and *prod₁*, and so on.

The next step consists of comparing, in *user-DB*, the user's and the producer's map versions, *user₁* and *prod₁*, for the detection of possible conflictual and non-conflictual updates.

3.4 Comparison of updates and detection of conflicts

When the system compares the values associated with *prod₁* and *user₁* for the different multiversion objects in the database, the two following situations are possible:

Case 1. An object has the same value in both map versions. This can be because the object has not been updated - i.e., the multiversion object contains only one value for *prod₀*, shared implicitly by *prod₁* and *user₁*. Alternatively this may happen when the object has been updated or created in both the *prod-DB* and *user-DB*, and the values are equal, e.g., road section *R₂* that has been deleted in map versions *user₁* and *prod₁*. In these cases, there is no conflict.

Case 2. The object has different values in the two map versions, *user₁* and *prod₁*. This situation is possible in the following cases:

- a) the object has been updated or created in both *user₁* and *prod₁*, and the two values are different; e.g., parcel *P₆* has value *valp6* in *prod₀*,

- valp6x* in *user₁*, and *valp6a* in *prod₁*, and *valp6x*, and *val6a* are different. Here, the two updates or creations are in conflict.
- b) the object has been updated either in *prod₁* only, or in *user₁* only. The update corresponds to one of these operations:
- a creation: roundabout *P₇* and road *R₆* have been created in map version *prod₁*, and antenna *A* in *user₁*.
 - a deletion: road *R₀* is deleted in *prod₁* and still exists in *user₁*.
 - an update of its value: road *R₅* has value *valr5a* in *prod₁* and value *valr5* in *user₁* (implicit sharing with *prod₀*).
 - a group operation: the value of parcel *P₂* in *user₁* is *valp2*, implicitly shared with *prod₀*. Its value in *prod₁* is denoted by #, meaning that a group operation, that is explained by the genealogy graph of *P₂*: it has been split into two new parcels *P_{2a}* and *P_{2b}*. These two parcels have only one value in *prod₁* (corresponding to their creation). Conflicts exist in these cases and, for each object, its value in the new map version to be created in the *user-DB* depends on the user's decision.

These different situations can be visualized on the map by using special coloring of the object, revealing non-conflictual and conflictual updates and the types of conflicts.

This section has focused on updates relating to objects only. For schema updates a similar procedure is adopted (Bellosta, Cellary & Jomier, 1998); e.g., antenna if it exists only in *user-DB*.

3.5 Proposed strategy for updates propagation

The previous steps of the UMV methodology help the user in the visual detection of conflicts both at schema and object levels. Moreover, it supplies information concerning the types of evolution underlying the different updates. Now, the remaining step concerns the propagation of the detected updates in the user database.

This step needs an appropriate strategy that may depend on many factors of the application concerned - e.g., knowledge about the underlying topology of the spatial objects (Egenhofer, Clementini & Di Felice, 1994; Badard & Lemarié, 1999). For instance, the user may decide whether to favor his update in place of the producer's one in case of conflict. This choice may impact the user's added information, which may need to be readjusted. It is, therefore, wiser to use already proposed strategies such as (Badard & Lemarié, 1999, Badard 2000, Kilpeläinen 1997, Uitermark *et al* 1998),

where the propagation problem has been deeply studied. The final map version of the user after the propagation of updates may contain the producer's updates as well as those of the user. Suppose that the final user's map version is *prod-user₁* as illustrated in Fig. 6.

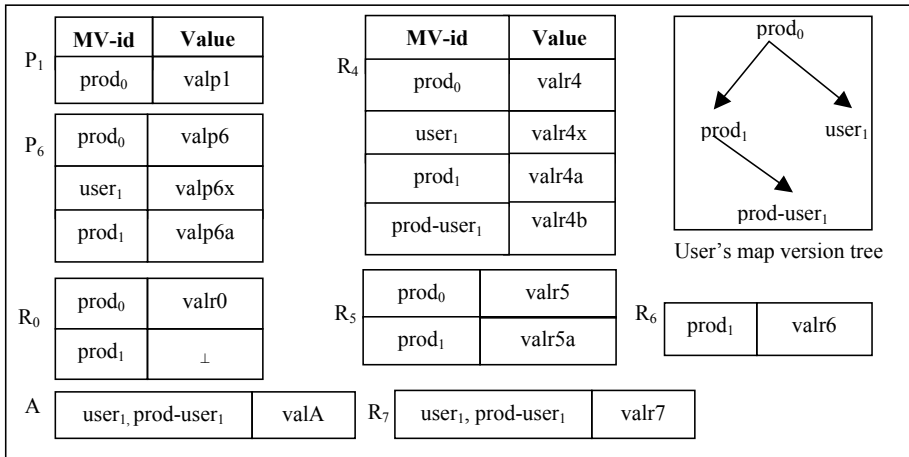


Fig. 6 - Part of user-DB after the integration of data from *prod₁*

In order to obtain the map version *prod-user₁*, the user first derives a map version identified as *prod-user₁* from *prod₁* (see Fig. 6), since *prod₁* contains a large part of updates that the user needs to represent in his map version. Thus, all objects in the new map version are shared implicitly with *prod₁*, e.g., parcels *P₁*, *P₆* and road sections *R₅*, *R₆*. Next, he includes in *prod-user₁* his specific updates: the road section *R₇* and the antenna *A*. Fig. 6 shows part of the state of *user-DB* after the creation of the final map version *prod-user₁*, resulting from the merging of *user₁* and *prod₁*. In the user's database, the value *valr7* of the road section *R₇* in *prod-user₁* is the value coming from *user₁*, and it is shared explicitly with *user₁* as illustrated in Fig. 6. The situation is the same for antenna *A* that the user preserves in *prod-user₁*. In this case, an object in *prod-user₁* has a value different from that in *prod₁* and *user₁*, for instance road section *R₄* in Fig. 6, the system creates a new entry for this value, *valr4b* that is associated with *prod-user₁*.

Finally, from *user-DB* of comprising of four map versions, the system reads the value of an object *O* in *prod-user₁*, as follows: if *prod-user₁* appears in the multiversion object, then the value *O* is the one corresponding to *prod-user₁* (see Fig. 6). Else, if *prod-user₁* does not appear in the multiversion object, then the value *O* is shared implicitly with the value of the nearest ancestor, obtained from the user's map version tree that is either *prod₁* (e.g., *P₆*, *R₆*), or *prod₀* (e.g., *P₁*); finally, if only *user₁* appears in the multiversion object, then *O* does not exist in *prod-user₁*.

After this propagation step, the user will work on new versions of *prod-user₁* for further updates, whereas the producer uses new versions of *prod₁*. Thus at time t_2 , $t_2 > t_1$, a new operation of integration of updates may take place: the new up-to-date map version, *prod₂*, is inserted into the user's database and compared to the current user's map version, *user₂*, which has been created by derivation and updates from *prod-user₁*. A new map version *prod-user₂* is created, integrating updates coming from *prod₂* and *user₂*. In order to help users in understanding updates performed in the different map versions for integration purposes, the updates should be documented as depicted in (Peerbocus *et al*, 2001).

4 Conclusions

The focus of this paper was on how to help the exchange of updating information between a geographic data producer and a user. The main advantage of the UMV methodology is that it allows the automatic detection of updates, whereas existing techniques require an exhaustive retrieval within the different versions of the database. The UMV methodology responds as well when updates are delivered on a given frequency as for real time updates. The UMV methodology can also be applied in a general context where there is a need of exchanging geographic data between any two users, or between a user and a producer.

A prototype of multiversion geographic database has been developed using MapInfo® in the LAMSADE Laboratory, University Paris-Dauphine. It requires the implementation of the version mechanism in the geographic database, which must be managed by a version manager. It allows the representation of the different states of geographic objects. All changes are documented. The prototype allows the retrieval of updated geographic objects between any two map versions of the multiversion geographic database and provides the user with the associated change documentation (Hedjar, 2001).

The integration of the updates and the propagation of their effects in geographic databases require handling all the spatial relationships between entities in order to preserve consistency or added information. Several research works have set up tools for the retrieval of these relationships necessary to the updating of geographic databases. Ongoing research (especially at IGN) investigate the set up of a formalism and a model for the design of geographic databases that are easier to maintain. The UMV methodology thus appears as a key element of a global methodology for the design of easy-to-update GIS.

References

- Badard T., 1998a. Towards a generic updating tool for geographic databases. GIS/LIS'98, Fort Worth, Texas, pp. 352-363.
- Badard T., 1998b. Extraction des mises à jour dans les Base de Données Géographiques. *Revue Int. de Géomatique*, vol. 8, 1-2, pp. 121-147.
- Badard T., 2000. Propagation des mises à jour dans les bases de données géographiques multi-représentations. PhD, Univ. Marne-la-Vallée, France.
- Badard T. and Lemarié C., 1999. Propagating updates between geographic databases with different scales. *Innov. in GIS VII:GeoComputation*, London.
- Badard T. and Richard D., 2001. Using XML for the exchange of updating information between GIS. *CEUS 25*. Elsevier, Oxford, pp. 17-31.
- Bauzer-Medeiros C. and Jomier G., 1993. Managing Alternatives and Data Evolution in GIS. *ACM Workshop on Advances in GIS*, Arlington, Virginia.
- Bellosta M.J., Cellary W. and Jomier G., 1998. Consistent Versioning of OODB Schema and its Extension. 14^e Journées BDA, Hammamet, Tunisia.
- Cellary W. and Jomier G., 1990. Consistency of versions in object-oriented databases. *VLDB*, Brisbane, pp. 432-441.
- Cellary W. and Jomier G., 2000. The Database Version Approach. *Networking and Information Systems Journal*, Hermès, Paris, Vol. 3, No 1, pp. 177-214.
- Devogèle T., 1998. Le processus d'intégration et d'appariement des BD géographiques. PhD, Univ. Versailles-Saint Quentin, France.
- Easterfield M.E., Newell R.G. and Theriault D.G., 1992. Version Management in GIS - Applications and Techniques. *Smallworld technical paper n° 4*.
- Egenhofer M.J., Clementini E. and Di Felice P., 1994. Evaluating inconsistencies among multiple representations. *SDH6th*, Edinburgh, UK, pp. 901-920.
- Gançarski S. and Jomier G., 1994. Managing Entity Versions within their Context: a Formal Approach. *DEXA'94*, Athens, LCNS n° 856, pp. 400-409.
- Hedjar M., 2001. A prototype for documenting spatiotemporal evolution. Research report, DEA 127, LAMSADE, Univ. Paris-Dauphine, France.

- International Hydrographic Organisation, 1996. "IHO transfer standard for digital hydrographic data", Publication S-57, Edition 3.0, 126 p.
- Kilpeläinen T., 1997. Multiple representation and generalisation of geo-databases for topographic maps. Finnish Geodetic Inst., 124, 51-711-212-4.
- Lemarié C., Raynal L., 1996. Geographic data matching: First investigations for a generic tool. GIS/LIS'96, Denver, Colorado, pp. 405-420.
- Peerbocus M.A., Bauzer Medeiros C., Jomier G. and Voisard A., 2001. Documenting Changes in a Spatiotemporal DB. XVI BSDB, Rio.
- Poupart-Lavoie G., 1997. Développement d'une méthode de transfert des mises à jour de données à référence spatiale. MSc, Université Laval, Québec, 128 p.
- Raynal L., 1996. Some elements for modelling updates in topographic database. GIS/LIS'96, Denver, Colorado, pp. 405-420.
- Sperry L., Claramunt C. and Libourel T., 1999. A Lineage Metadata Model for the Temporal Management of a Cadastre Application. Int. Workshop on Spatio-Temporal Models and Languages, Firenze, Italy.
- Uitermark H. *et al*, 1998. Propagating updates: Finding Corresponding objects in a multi-source environment. 8th SDH, Vancouver, pp. 580-591.

A Characterisation of PQI Interval Orders*

Alexis Tsoukiàs[†], Philippe Vincke[‡]

Résumé

Dans ce papier nous présentons la solution à un problème ouvert concernant la représentation de préférences sur des intervalles. Étant donné un ensemble et trois relations binaires sur celui-ci (indifférence, préférence faible, préférence stricte) nous présentons les conditions nécessaires et suffisantes pour pouvoir associer à chaque élément de l'ensemble un intervalle de façon à: 1) obtenir une indifférence si les deux intervalles sont inclus l'un dans l'autre; 2) obtenir une préférence faible si un intervalle "est plus à droite que l'autre", mais les deux intervalles ont une intersection non vide; 3) obtenir une préférence stricte si les deux intervalles sont disjoints et qu'un intervalle "est plus à droite que l'autre".

Mots-clefs : Intervals, Ordres d'Intervalle, Indifférence, Préférence Faible, Préférence Stricte

Abstract

We provide an answer to an open problem concerning the representation of preferences by intervals. Given a finite set of elements and three relations on this set (indifference, weak preference and strict preference), necessary and sufficient conditions are provided for representing the elements of the set by intervals in such a way that 1) two elements are indifferent when the interval associated to one of them is included in the interval associated to the other; 2) an element is weakly preferred to another when the interval of the first is "more to the right" than the interval of the other, but the two intervals have a non empty intersection; 3) an element is strictly preferred to another when the interval of the first is "more to the right" than the interval of the other and their intersection is empty.

Key words : Intervals, Interval Orders, Indifference, Weak Preference, Strict Preference

* Preliminary version of a text in press with Discrete Applied Mathematics

[†] LAMSADE - CNRS, Université Paris-Dauphine, 75775 Paris Cedex 16,
tsoukias@lamsade.dauphine.fr

[‡] SMG - ULB, CP 210/01, Bld du Triomphe, 1050 Bruxelles, pvincke@ulb.ac.be

1 Introduction

Comparing intervals is a frequently encountered problem in preference modelling and decision aid. This is due to the fact that the comparison of alternatives (outcomes, objects, candidates, ...) generally are realised through their evaluations on numerical scales, while such evaluations often are imprecise or uncertain. A well known preference structure, in this context, is the semi order (see Luce, 1956 and for a comprehensive presentation Pirlot and Vincke, 1997) and more generally the interval order (see also Fishburn, 1985). An interval order is obtained when one considers that an alternative is preferred to another iff it's interval is "completely to the right" of the other (hereafter we assume that the larger an evaluation of an alternative is on a numerical scale the better the alternative is), while any two alternatives the intervals of which have a non empty intersection are considered indifferent. Such a model has a strict probabilistic interpretation, since the intervals associated to each alternative can be viewed as the extremes of the probability distributions of the evaluations of the alternatives. Under such an interpretation a "sure preference" occurs only if the distributions have an empty intersection. A second implicit assumption in this frame is that if there is no preference of an alternative over the other then they are indifferent.

It is easy however to notice that if, in the previous frame, we want to establish a "sure indifference", it is much more natural to consider that two alternatives are indifferent if their associated intervals (or distributions) are embedded. In such a case we obtain a preference relation which is known to be a partial order of dimension 2 (a partial order obtained from the intersection of exactly two linear orders; see Roubens and Vincke, 1985).

Practically we observe that we have three situations:

- a "sure indifference": when the intervals associated to two alternatives are embedded;
- a "sure preference": when the interval associated to one alternative is "more to the right" with respect to the interval associated to the other alternative and the two intervals have an empty intersection;
- an "hesitation between indifference and preference" which we denote as weak preference: when the interval associated to one alternative is "more to the right" with respect to the interval associated to the other alternative and the two intervals have a non empty intersection.

Such an interpretation fits better in the case we have qualitative uncertainties or imprecision and is consistent with the use of specific relations in order to represent situations of hesitation in preference modelling (see Tsoukiàs and Vincke, 1997). However, such a preference structure (hereafter called *PQI* interval order) lacked any characterisation as mentioned for instance in Vincke, 1988 (by characterisation we mean the determination of a list of properties concerning the three preference relations which are necessary and sufficient conditions in order to be able to represent them by intervals as mentioned

before).

In this paper we present an answer for this problem. Section 2 provides the basic notations and definitions. In section 3 we recall some results concerning conventional interval orders. The main result is presented, demonstrated and discussed in section 4. Finally section 5 presents an algorithm for the detection of a *PQI* interval order on a set A .

2 Notations and Definitions

In this paper we consider binary relations defined on a finite set A , that is subsets of $A \times A$ (the quantifiers apply therefore always to such a domain). Further on we will use the following notations for any binary relations S, T . If S is a binary relation on A we denote by $S(x, y)$ the fact that $(x, y) \in S$. \neg, \wedge and \vee denote the usual negation, conjunction and disjunction operations.

$$\begin{aligned} S^{-1} &= \{(x, y) : S(y, x)\} \\ S^c &= \neg S = \{(x, y) : \neg S(x, y)\} \\ S^d &= \neg S^{-1} = \{(x, y) : \neg S(y, x)\} \\ S \subset T &: \forall x, y \ S(x, y) \rightarrow T(x, y) \\ S.T &= \{(x, y) : \exists z \ S(x, z) \wedge T(z, y)\} \\ S^2 &= \{(x, y) : \exists z \ S(x, z) \wedge S(z, y)\} \\ S \cup T &= \{(x, y) : S(x, y) \vee T(x, y)\} \\ S \cap T &= \{(x, y) : S(x, y) \wedge T(x, y)\} \end{aligned}$$

We recall some well known definitions from the literature (our terminology follows Roubens and Vincke, 1985).

Definition 2.1 A relation S on a set A is said to be:

- reflexive: iff $\forall x \ S(x, x)$
- irreflexive: iff $\forall x \ \neg S(x, x)$
- symmetric: iff $\forall x, y \ S(x, y) \rightarrow S^{-1}(x, y)$
- asymmetric: iff $\forall x, y \ S(x, y) \rightarrow S^d(x, y)$
- complete: iff $\forall x, y, \ x \neq y, \ S(x, y) \vee S^{-1}(x, y)$
- transitive: iff $\forall x, y, z \ S(x, y) \wedge S(y, z) \rightarrow S(x, z)$
- negatively transitive: iff $\forall x, y, z \ \neg S(x, y) \wedge \neg S(y, z) \rightarrow \neg S(x, z)$

Definition 2.2 A binary relation S is:

- a partial order iff it is asymmetric and transitive;

- a weak order iff it is asymmetric and negatively transitive;
- a linear order iff it is irreflexive, complete and transitive;
- an equivalence iff it is reflexive, symmetric and transitive.

In this paper we will consider relations representing strict preference, weak preference and indifference situations. We will denote them P, Q, I respectively. Moreover, such relations are expected to satisfy some “natural” properties of the type announced in the following two definitions.

Definition 2.3 A $\langle P, I \rangle$ preference structure on a set A is a couple of binary relations, defined on A , such that:

- I is reflexive and symmetric;
- P is asymmetric;
- $I \cup P$ is complete;
- P and I are mutually exclusive ($P \cap I = \emptyset$).

Definition 2.4 A $\langle P, Q, I \rangle$ preference structure on a set A is a triple of binary relations, defined on A , such that:

- I is reflexive and symmetric;
- P and Q are asymmetric;
- $I \cup P \cup Q$ is complete;
- P, Q and I are mutually exclusive.

Finally we introduce an equivalence relation as follows:

Definition 2.5 The equivalence relation associated to a $\langle P, Q, I \rangle$ preference structure is the binary relation E , defined on the set A , such that, $\forall x, y \in A$:

$$E(x, y) \text{ iff } \forall z \in A : \begin{cases} P(x, z) \Leftrightarrow P(y, z) \\ Q(x, z) \Leftrightarrow Q(y, z) \\ I(x, z) \Leftrightarrow I(y, z) \\ Q(z, x) \Leftrightarrow Q(z, y) \\ P(z, x) \Leftrightarrow P(z, y) \end{cases}$$

Remark 2.1 In this paper we consider that two different elements of A are never equivalent for the given $\langle P, Q, I \rangle$ preference structure. This is not restrictive as it suffices to consider the quotient of A by E to satisfy the assumption. Under such an assumption we will use in the numerical representation of the preference relations only strict inequalities without any loss of generality.

3 Interval Orders

In this section we recall some definitions and theorems concerning conventional interval orders and semi orders.

Definition 3.1 A $\langle P, I \rangle$ preference structure on a set A is a *PI interval order* iff $\exists l, r : A \mapsto \mathcal{R}^+$ such that:
 $\forall x : r(x) > l(x)$
 $\forall x, y : P(x, y) \Leftrightarrow l(x) > r(y)$
 $\forall x, y : I(x, y) \Leftrightarrow l(x) < r(y) \text{ and } l(y) < r(x)$

Definition 3.2 A $\langle P, I \rangle$ preference structure on a set A is a *PI semi order* iff $\exists l : A \mapsto \mathcal{R}^+$ and a positive constant k such that:
 $\forall x, y : P(x, y) \Leftrightarrow l(x) > l(y) + k$
 $\forall x, y : I(x, y) \Leftrightarrow |l(x) - l(y)| < k$

Such structures have been extensively studied in the literature (see for example Fishburn, 1985). We recall here below the two fundamental results which characterize interval orders and semi orders.

Theorem 3.1 A $\langle P, I \rangle$ preference structure on a set A is a *PI interval order* iff $P.I.P \subset P$.

Proof. See Fishburn, 1985.

Theorem 3.2 A $\langle P, I \rangle$ preference structure on a set A is a *PI semi order* iff $P.I.P \subset P$ and $I.P.P \subset P$.

Proof. See Fishburn, 1985.

4 $\langle P, Q, I \rangle$ Interval Orders

As mentioned in the introduction, we are interested in situations where, comparing elements evaluated by intervals, one wants to distinguish three situations: indifference if one interval is included in the other, strict preference if one interval is completely “to the right” of the other and weak preference when one interval is “to the right” of the other, but they have a non empty intersection. Definition 4.1 precisely states this kind of situation, $l(x)$ and $r(x)$ respectively representing the left and right extremities of the interval associated to any element $x \in A$.

Definition 4.1 A $\langle P, Q, I \rangle$ preference structure on a finite set A is a PQI interval order, iff there exist two real valued functions l and r such that, $\forall x, y \in A, x \neq y$:

- $r(x) > l(x)$;
- $P(x, y) \Leftrightarrow r(x) > l(x) > r(y) > l(y)$;
- $Q(x, y) \Leftrightarrow r(x) > r(y) > l(x) > l(y)$;
- $I(x, y) \Leftrightarrow r(x) > r(y) > l(y) > l(x)$ or $r(y) > r(x) > l(x) > l(y)$.

The reader will notice that the above definition immediately follows Definition 3.1, since a preference structure characterised as a PI interval order can always be seen as a PQI interval order also. We give now necessary and sufficient conditions for which such a preference structure exists.

Theorem 4.1 A $\langle P, Q, I \rangle$ preference structure on a finite set A is a PQI interval order, iff there exists a partial order I_l such that:

- i) $I = I_l \cup I_r \cup I_o$ where $I_o = \{(x, x), x \in A\}$ and $I_r = I_l^{-1}$;
- ii) $(P \cup Q \cup I_l)P \subset P$;
- iii) $P(P \cup Q \cup I_r) \subset P$;
- iv) $(P \cup Q \cup I_l)Q \subset P \cup Q \cup I_l$;
- v) $Q(P \cup Q \cup I_r) \subset P \cup Q \cup I_r$;

Proof.

We first give an outline of necessity demonstration which is the easy part of the theorem. If $\langle P, Q, I \rangle$ is a PQI interval order, then defining

- $I_l(x, y) \Leftrightarrow l(y) < l(x) < r(x) < r(y)$
- $I_r(x, y) \Leftrightarrow l(x) < l(y) < r(y) < r(x)$

we obtain two partial orders satisfying the desired properties. As an example we demonstrate property (v):

$Q(x, y)$ and $(P \cup Q \cup I_r)(y, z)$ imply $r(x) > r(y)$ and $r(y) > r(z)$, hence $r(x) > r(z)$, so that $(P \cup Q \cup I_r)(x, z)$.

Conversely let us assume the existence of I_l satisfying the properties of the theorem. Define a set A' isomorphic to A and denote by x' the image of $x \in A$ in A' . In the set $A \cup A'$ let us define the relation S as follows: $\forall x, y \in A, x \neq y$

- $S(x', x)$
- $S(x, y) \Leftrightarrow (P \cup Q \cup I_l)(x, y)$
- $S(x', y') \Leftrightarrow (P \cup Q \cup I_r)(x, y)$
- $S(x, y') \Leftrightarrow P(x, y)$
- $S(x', y) \Leftrightarrow \neg P(y, x)$

We demonstrate now that S is a linear order (irreflexive, complete and transitive relation) in $A \cup A'$.

Irreflexivity results from irreflexivity of P, Q, I_l and I_r .

To demonstrate completeness of S remark that for $x \neq y$:

$$\begin{aligned} \neg S(x, y) &\Leftrightarrow \neg(P \cup Q \cup I_l)(x, y) \\ &\Leftrightarrow (P \cup Q \cup I_l)(y, x) \text{ since } P \cup Q \cup I \text{ is complete and } I = I_l \cup I_r \cup I_o \\ &\Leftrightarrow S(y, x) \end{aligned}$$

$$\begin{aligned} \neg S(x', y') &\Leftrightarrow \neg(P \cup Q \cup I_r)(x, y) \\ &\Leftrightarrow (P \cup Q \cup I_r)(y, x) \text{ since } P \cup Q \cup I \text{ is complete and } I = I_l \cup I_r \cup I_o \\ &\Leftrightarrow S(y', x') \end{aligned}$$

$$\begin{aligned} \neg S(x, y') &\Leftrightarrow \neg P(x, y) \\ &\Leftrightarrow S(y', x) \end{aligned}$$

$$\begin{aligned} \neg S(x', y) &\Leftrightarrow P(y, x) \\ &\Leftrightarrow S(y, x') \end{aligned}$$

We demonstrate now that S is transitive.

- $S(x, y)$ and $S(y, z)$ imply $(P \cup Q \cup I_l)(x, y)$ and $(P \cup Q \cup I_l)(y, z)$. From conditions ii) and iv) of the theorem, we know that $(P \cup Q \cup I_l)(x, y)$ and $(P \cup Q)(y, z)$ imply $(P \cup Q \cup I_l)(x, z)$, hence $S(x, z)$. From transitivity of I_l we have that $I_l(x, y)$ and $I_l(y, z)$ imply $I_l(x, z)$, hence $S(x, z)$. Finally, if $(P \cup Q)(x, y)$ and $I_l(y, z)$ then $(P \cup Q \cup I_l)(x, z)$ because, if not, we would have $(P \cup Q \cup I_l)(z, x)$ which with $I_l(y, z)$ would give $(P \cup Q \cup I_l)(y, x)$ (by conditions ii) and iv) and transitivity of I_l), contradiction. So we get $S(x, z)$.
- $S(x, y)$ and $S(y, z')$ imply $(P \cup Q \cup I_l)(x, y)$ and $P(y, z)$, which, by condition ii), give $P(x, z)$, hence $S(x, z')$.
- $S(x, y')$ and $S(y', z)$ imply $P(x, y)$ and $\neg P(z, y)$. If $\neg S(x, z)$, then $(P \cup Q \cup I_l)(z, x)$ which, with $P(x, y)$ and by condition ii) would give $P(z, y)$, a contradiction. Thus $S(x, z)$. This reasoning applies also in the case $y = z$.
- $S(x, y')$ and $S(y', z')$ imply $P(x, y)$ and $(P \cup Q \cup I_r)(y, z)$, which, by condition iii), give $P(x, z)$, hence $S(x, z')$.

- $S(x', y')$ and $S(y', z)$ imply $(P \cup Q \cup I_r)(x, y)$ and $\neg P(z, y)$. If $\neg S(x', z)$, then $P(z, x)$ which, with $(P \cup Q \cup I_r)(x, y)$ and by condition iii) would give $P(z, y)$, a contradiction. Thus $S(x', z)$. This reasoning applies also in the case $y = z$.
- $S(x', y')$ and $S(y', z')$ imply $(P \cup Q \cup I_r)(x, y)$ and $(P \cup Q \cup I_r)(y, z)$. From conditions iii) and v) of the theorem, we know that $(P \cup Q)(x, y)$ and $(P \cup Q \cup I_r)(y, z)$ imply $(P \cup Q \cup I_r)(x, z)$, hence $S(x', z')$. From transitivity of I_r we have that $I_r(x, y)$ and $I_r(y, z)$ imply $I_r(x, z)$, hence $S(x', z')$. Finally, if $I_r(x, y)$ and $(P \cup Q)(y, z)$ then $(P \cup Q \cup I_r)(x, z)$ because, if not, we would have $(P \cup Q \cup I_r)(z, x)$ which with $I_r(x, y)$ would give $(P \cup Q \cup I_r)(z, y)$ (by condition iii) and v) and transitivity of I_r), contradiction. So we get $S(x', z')$.
- $S(x', y)$ and $S(y, z)$ imply $\neg P(y, x)$ and $(P \cup Q \cup I_l)(y, z)$ If $\neg S(x', z)$, then $P(z, x)$ which, with $(P \cup Q \cup I_l)(y, z)$ and by condition ii) would give $P(y, x)$, a contradiction. Thus $S(x', z)$. This reasoning applies also in the case $y = x$.
- $S(x', y)$ and $S(y, z')$ imply $\neg P(y, x)$ and $P(y, z)$. If $\neg S(x', z')$, then $(P \cup Q \cup I_r)(z, x)$ which, with $P(y, z)$ and by condition iii) would give $P(y, x)$, a contradiction. Thus $S(x', z')$. This reasoning applies also in the case $y = x$.

Since S is a linear order on $A \cup A'$, there exists a real valued function u such that,
 $\forall x, y \in A$:

- $S(x, y) \Leftrightarrow u(x) > u(y)$;
- $S(x', y') \Leftrightarrow u(x') > u(y')$;
- $S(x, y') \Leftrightarrow u(x) > u(y')$;
- $S(x', y) \Leftrightarrow u(x') > u(y)$.

We define $\forall x \in A$, $l(x) = u(x)$ and $r(x) = u(x')$ and we obtain:

- $\forall x : r(x) > l(x)$, since $S(x', x)$.
- $\forall x, y : P(x, y) \Leftrightarrow S(x, y') \Leftrightarrow l(x) > r(y)$.
- $\forall x, y : Q(x, y) \Leftrightarrow S(x, y) \wedge S(x', y') \wedge \neg P(x, y) \Leftrightarrow$
 $l(x) > l(y)$ and $r(x) > r(y)$ and $r(y) > l(x)$, equivalent to:
 $r(x) > r(y) > l(x) > l(y)$.
- $\forall x, y : I(x, y) \Leftrightarrow$
 $r(x) > r(y) > l(y) > l(x)$ or $r(y) > r(x) > l(x) > l(y)$
 since $I(x, y)$ holds in all the remaining cases.



We can complete the investigation providing a characterisation of PQI semi orders.

Definition 4.2 A PQI semi order is a PQI interval order such that $\exists k > 0$ constant for which $\forall x : r(x) = l(x) + k$

In other words, a PQI semi order is a $\langle P, Q, I \rangle$ preference structure for which there exists a real valued function $l : A \mapsto \mathcal{R}$ and a positive constant k such that $\forall x, y$:

- $P(x, y) \Leftrightarrow l(x) > l(y) + k$;
- $Q(x, y) \Leftrightarrow l(y) + k > l(x) > l(y)$;
- $I(x, y) \Leftrightarrow l(x) = l(y)$; (in fact I reduces to I_o).

For such preference structures the following theorem holds.

Theorem 4.2 A $\langle P, Q, I \rangle$ preference structure is a PQI semi order iff:

- i) I is transitive
- ii) $PP \cup PQ \cup QP \subset P$;
- iii) $QQ \subset P \cup Q$;

Proof

Necessity is trivial. We give only the sufficiency proof. Since I is an equivalence relation, we consider the relation $P \cup Q$ on the set A/I . Such a relation is clearly a linear order (irreflexivity and completeness result from definition 2.4 and transitivity from conditions ii) and iii) of the theorem). Therefore we can index the elements of A/I by $i = 1, 2 \dots n$ in such a way that $\forall x_i, x_{i+1} \in A/I : (P \cup Q)(x_{i+1}, x_i)$.

Choosing an arbitrary positive value k , we define function l as follows:

- $l(x_1) = 0$ and for $i = 2, 3, \dots n$
- $l(x_{i+1}) > l(x_i)$
- $l(x_i) > l(x_j) + k \quad \forall j < i$ such that $P(x_i, x_j)$
- $l(x_i) < l(x_m) + k \quad \forall m < i$ such that $Q(x_i, x_m)$.

This is always possible because $P(x_i, x_j)$ and $Q(x_i, x_m)$ imply $(P \cup Q)(x_m, x_j)$ (if not, we would have $(P \cup Q)(x_j, x_m)$ which, with $P(x_i, x_j)$ and by condition ii) would give $P(x_i, x_m)$, hence $m > j$ and $l(x_m) > l(x_j)$). By construction the function l satisfies the numerical representation of a PQI semi order.



5 Detection of a PQI Interval Order

The problem is the following:

Given a set A and a $\langle P, Q, I \rangle$ preference structure on it, verify whether it is a PQI interval order. The difficulty resides in the fact that the theorem previously announced contains a second order condition which is the existence of the partial order I_l . For this purpose we give two propositions which show the difficulties in detecting such a structure.

Proposition 5.1 *There exist $\langle P, Q, I \rangle$ preference structures which are $P\hat{I}$ -interval orders (where $\hat{I} = Q \cup I \cup Q^{-1}$), but are not PQI interval orders.*

Proof Consider the following case.

- $A = \{a, b, c, d, e\}$;
- $P = \{(a, c), (d, e), (a, e)\}$;
- $Q = \{(d, c), (a, b), (b, e)\}$;
- $I = \{(a, d), (c, e), (b, d), (b, c), (d, a), (e, c), (d, b), (c, b)\} \cup I_o$

On the one hand if we consider the relation $\hat{I} = Q \cup I \cup Q^{-1}$ it is easy to observe that the $\langle P, \hat{I} \rangle$ preference structure is a PI interval order ($P\hat{I}P \subset P$ holds). On the other hand if we accept that the given $\langle P, Q, I \rangle$ preference structure is a PQI interval order then we have (by the definition 4.1 and the theorem 4.1) that:

- $I(a, d)$ has to be $I_l(a, d)$ because of c ;
- $I(d, b)$ has to be $I_l(d, b)$ because of e ;

therefore by transitivity we should have $I_l(a, b)$, while we have $Q(a, b)$ which is impossible. Therefore we can conclude that for this particular case the PQI interval order representation is impossible. ■

Proposition 5.2 *There exist $\langle P, Q, I \rangle$ preference structures which have more than one PQI interval order representation.*

Proof Consider the following case.

- $A = \{a, b, c\}$;
- $P = \emptyset$;
- $I = \{(a, c), (b, c), (c, a), (c, b)\} \cup I_o$;
- $Q = \{(a, b)\}$

It is easy to observe that both $I_l(a, c), I_l(b, c)$ and $I_l(c, a), I_l(c, b)$ are possible, thus allowing two different PQI interval orders: one in which the interval of c is included in

the intervals of both a and b and the other where the intervals of b and a are included in the interval c . Both representations are correct, although incompatible with each other. ■

In order to detect if a $\langle P, Q, I \rangle$ preference structure is a PQI interval order we propose the following algorithm which we present in terms of pseudo-code.

Step 1 For all x, y verify that $P^2 \subset P, P.Q \subset P, Q.P \subset P$ and $Q^2 \subset P \cup Q$.

Step 2 $\forall x, y, z \ I(x, y) \wedge P(x, z) \wedge Q(y, z) \rightarrow I_l(x, y)$

Step 3 $\forall x, y, z \ I(x, y) \wedge P(z, x) \wedge Q(z, y) \rightarrow I_l(x, y)$

Step 4 $\forall x, y, z \ I(x, y) \wedge I(y, z) \wedge P(x, z) \rightarrow I_l(x, y) \wedge I_l(z, y)$

Step 4 bis $\forall x, y, z \ I(x, y) \wedge I(y, z) \wedge Q(x, z) \rightarrow (I_l(x, y) \wedge I_l(z, y)) \vee (I_l(y, x) \wedge I_l(y, z))$

Step 5 $\forall x, y, z \ I_l(x, y) \wedge I_l(y, z) \rightarrow I_l(x, z)$

Step 6 For a x, y such that $I(x, y)$ and I_l has not been established, choose arbitrary $I_l(x, y)$ and go to step 5.

The algorithm succeeds if it arrives to assign all elements of relation I to the relation I_l or to the relation I_r without any contradiction, that is without assigning to a relation a couple already assigned to another relation.

Proposition 5.3 *If the above algorithm succeeds, then the $\langle P, Q, I \rangle$ preference structure is a PQI interval order.*

Proof

We have to demonstrate that the conditions of Theorem 4.1 are verified.

1. Exists a partial order I_l such that $I = I_l \cup I_o \cup I_l^{-1}$. By construction of I_l .
2. $(P \cup Q \cup I_l).P \subset P$.
 $P.P \subset P$ by step 1;
 $Q.P \subset P$ by step 1;
 $I_l.P \subset P$. Suppose that:
 $\exists x, y, z : I_l(x, y) \wedge P(y, z) \wedge P(z, x)$.
 Impossible since it implies $P(y, x)$ step 1
 $\exists x, y, z : I_l(x, y) \wedge P(y, z) \wedge Q(z, x)$.

Impossible since it implies $P(y, x)$ step 1

$\exists x, y, z : I_l(x, y) \wedge P(y, z) \wedge I_l(z, x)$.

Impossible since it implies $I_l(z, y)$ step 5

$\exists x, y, z : I_l(x, y) \wedge P(y, z) \wedge I_l(x, z)$.

Impossible since it implies $P(z, y)$ step 4

$\exists x, y, z : I_l(x, y) \wedge P(y, z) \wedge Q(x, z)$.

Impossible since it implies $I_l(y, x)$ step 2.

3. $P.(P \cup Q \cup I_l^{-1}) \subset P$.

$P.P \subset P$ by step 1;

$P.Q \subset P$ by step 1;

$P.I_l^{-1} \subset P$. Suppose that:

$\exists x, y, z : P(x, y) \wedge I_l^{-1}(y, z) \wedge P(z, x)$.

Impossible since it implies $P(z, y)$ step 1

$\exists x, y, z : P(x, y) \wedge I_l^{-1}(y, z) \wedge Q(z, x)$.

Impossible since it implies $P(y, x)$ step 1

$\exists x, y, z : P(x, y) \wedge I_l^{-1}(y, z) \wedge I_l(z, x)$.

Impossible since it implies $P(y, x)$ step 4

$\exists x, y, z : P(x, y) \wedge I_l^{-1}(y, z) \wedge I_l(x, z)$.

Impossible since it implies $I_l(x, y)$ step 5

$\exists x, y, z : P(x, y) \wedge I_l^{-1}(y, z) \wedge Q(x, z)$.

Impossible since it implies $I_l(y, z)$ step 3.

4. $(P \cup Q \cup I_l).Q \subset P \cup Q \cup I_l$.

$P.Q \subset P$ by step 1;

$Q.Q \subset P \cup Q$ by step 1;

$I_l.Q \subset P \cup Q \cup I_l$. Suppose that:

$\exists x, y, z : I_l(x, y) \wedge Q(y, z) \wedge P(z, x)$.

Impossible since it implies $P(y, x)$ step 1

$\exists x, y, z : I_l(x, y) \wedge Q(y, z) \wedge Q(z, x)$.

Impossible since it implies $P(y, x) \vee Q(y, x)$ step 1

$\exists x, y, z : I_l(x, y) \wedge Q(y, z) \wedge I_l(z, x)$.

Impossible since it implies $I_l(z, y)$ step 5.

5. $Q.(P \cup Q \cup I_l^{-1}) \subset P \cup Q \cup I_l^{-1}$.

$Q.P \subset P$ by step 1;

$Q.Q \subset P \cup Q$ by step 1;

$Q.I_l^{-1} \subset P \cup Q \cup I_l^{-1}$. Suppose that:

$\exists x, y, z : Q(x, y) \wedge I_l^{-1}(y, z) \wedge P(z, x)$.

Impossible since it implies $P(z, y)$ step 1

$\exists x, y, z : Q(x, y) \wedge I_l^{-1}(y, z) \wedge Q(z, x)$.

Impossible since it implies $P(y, x) \vee Q(y, x)$ step 1

$\exists x, y, z : Q(x, y) \wedge I_l^{-1}(y, z) \wedge I_l(x, z).$
 Impossible since it implies $I_l(x, y)$ step 5. ■

How difficult is it to verify whether a PQI preference structure is a PQI interval order? In other terms, what is the complexity of the previous algorithm? The reader may notice that in Step 6 we make an arbitrary choice. If after such a choice the algorithm reaches a contradiction normally we have to backtrack and try with a new choice. Actually we have a tree structure defined by the branches created by each arbitrary choice. The exploration of such a tree normally is in NP. However, our conjecture is that the introduction of Step 4bis (which is useless for the demonstration of the correctness of the algorithm) reduces the complexity of the algorithm to polynomial time, since a failure (reaching a contradiction) will be independent from any arbitrary choice previously done. This is the subject of a forthcoming paper (see also Ngo The, 1998).

Acknowledgements

An earlier version of this paper was presented in OSDA 98 and was thoroughly improved thanks to the comments of A. Dress. M. Pirlot suggested the example in Proposition 2. Two anonymous referees made also several valuable suggestions.

References

- [1] Fishburn P.C., *Interval Orders and Interval Graphs*, J. Wiley, New York, 1985.
- [2] Luce R.D., "Semiordeurs and a theory of utility discrimination", *Econometrica*, vol. 24, 1956, 178 - 191.
- [3] Ngo The A., "Algorithmes de detection d'ordres d'intervalle PQI", DEA Thesis, LAMSADE, Université Paris Dauphine, Paris, 1998.
- [4] Pirlot M., Vincke Ph., *Semi Orders*, Kluwer Academic, Dordrecht, 1997.
- [5] Roubens M., Vincke Ph., *Preference Modeling*, Springer Verlag, Berlin, 1985.
- [6] Tsoukiàs A., Vincke Ph., "Extended preference structures in MCDA", in J. Clímaco (ed.), *Multicriteria Analysis*, Springer Verlag, Berlin, 1997, 37 - 50.
- [7] Vincke Ph., "P,Q,I preference structures", in J. Kacprzyk, M. Roubens, eds., *Non conventional preference relations in decision making*, LNEMS 301, Springer Verlag, Berlin, 1988, 72 - 81.

A Hypocoloring Model for Batch Scheduling

D. de Werra^{*}, M. Demange[†], J. Monnot[‡], V. Th. Paschos[‡]

Résumé

Nous considérons un problème de sous-coloration pondérée dans un graphe modélisant des problèmes d'ordonnancement par lots; chaque sommet v a un poids $w(v)$; chaque couleur S est une collection de cliques disjointes deux à deux (au sens des sommets et des arêtes). Le poids $w(S)$ est défini comme étant $\max\{w(K) = \sum_{v \in K} w(v) \mid K \text{ clique} \in S\}$. Dans ce problème d'ordonnancement, le temps d'exécution est donné par $\sum_{i=1}^k w(S_i)$ où $S = (S_1, \dots, S_k)$ est une partition des sommets du graphe G dont chaque classe de couleur est définie comme précédemment. Nous présentons des propriétés de telles colorations concernant des classes particulières de graphes (line-graphes de cactus, block graphes) et nous exposons des résultats de complexité et d'approximabilité. Nous démontrons que le problème de décision associé est **NP-complet** pour deux classes de graphes : les graphes bipartis de degré maximum au plus 39 et les graphes planaires sans triangle de degré maximum au moins 3. Nous proposons également des algorithmes polynomiaux pour les graphes de degré maximum au plus 2 et pour les forêts de degré maximum au plus k , pour tout k constant. Finalement, nous présentons un algorithme exponentiel basé sur un principe de séparations pour les graphes sans triangle.

Mots-clefs : Ordonnancement par lots; Coloration; Sous-coloration; Hypocoloration; Coloration pondérée; Approximation

Abstract

Starting from a batch scheduling problem, we consider a weighted subcoloring in a graph G ; each node v has a weight $w(v)$; each color class S is a subset of nodes which generates a collection of node disjoint cliques. The weight $w(S)$ is defined as $\max\{w(K) = \sum_{v \in K} w(v) \mid K \in S\}$. In the scheduling problem, the

^{*} Ecole Polytechnique Fédérale de Lausanne, Switzerland, dewerra@dma.epfl.ch

[†] ESSEC, Dept. SID, France, demange@essec.fr

[‡] LAMSADE, CNRS UMR 7024, Université Paris-Dauphine, 75775 Paris Cedex 16, France, {monnot, paschos}@lamsade.dauphine.fr