INSTYTUT BADAŃ SYSTEMOWYCH
POLSKIEJ AKADEMII NAUK

TECHNIKI INFORMACYJNE TEORIA I ZASTOSOWANIA

Wybrane problemy
Tom 1(13)

poprzednio

ANALIZA SYSTEMOWA W FINANSACH I ZARZĄDZANIU

Pod redakcją
Jerzego HOŁUBCA

Warszawa 2011

INSTYTUT BADAŃ SYSTEMOWYCH
POLSKIEJ AKADEMII NAUK

TECHNIKI INFORMACYJNE TEORIA I ZASTOSOWANIA

Wybrane problemy
Tom 1(13)

poprzednio

ANALIZA SYSTEMOWA W FINANSACH I ZARZĄDZANIU

Pod redakcją
Jerzego HOŁUBCA

Warszawa 2011

Wykaz opiniodawców artykułów zamieszczonych
w niniejszym tomie:

Dr hab. inż. Przemysław GRZEGORZEWSKI, prof. PAN

Prof. dr hab. inż. Jerzy HOŁUBIEC

Dr inż. Tatiana JAWORSKA

Dr hab. inż. Wiesław KRAJEWSKI, prof. PAN

Dr hab. inż. Maciej KRAWCZAK, prof. PAN

Dr hab. Michał MAJSTEREK

Dr hab. inż. Andrzej MYŚLIŃSKI, prof. PAN

Prof. dr hab. inż. Witold PEDRYCZ

Dr hab. inż. Ryszard SMARZEWSKI, prof. KUL

Prof. dr hab. inż. Andrzej STRASZAK

Dr Dominik ŚLĘZAK

Prof. dr hab. inż. Stanisław WALUKIEWICZ

ISBN   9788389475336

IDENTIFICATION WITH COMPOUND OBJECT COMPARATORS – TECHNICAL ASPECTS

Łukasz Sosnowski
Studia Doktoranckie IBS PAN

In this paper I describe an implementation of compound object comparator to the problem of identification. I focus on technical aspects and testing of few final selecting method. This article shows some results of tests. In whole paper I show main application of an algorithm which was implemented in Polish Self-Government elections in 2010.

***Key words**: object identification, comparators, fuzzy sets, database, data warehouse, ICE, java*

1. Introduction

This article provides an example implementation of an algorithm identification of compound objects by using the structures called comparators. Types of compound objects have been defined earlier in article [6]. In this case we deal with the identification of geographical areas in the form of images. The theory for algorithm was presented in paper [7] which was used in visualization system of Local Government elections in Poland in 2010 with subsequent changes and modifications.

The algorithm was developed for the identification of geographic areas such counties and communities on the contour map of Poland using the available set of reference images consisting of the areas identified. The visualization of the results of the attendance on the map of Poland using a color scale was the final application.

The purpose of this part is to show the visualization results for the region in a global context - the whole of Poland. An example can be viewed at http://wybory2010.pkw.gov.pl/att/1/pl/000000.html # tabs-1

In this article, I wish to focus more attention to the implementation and some technical aspects. Chapter 2 presents a summary description of the algorithm described earlier in [7], with minor amendments, Chapter 3 presents the most interesting parts of the implementation of phase I and II of the algorithm. Chapter 4 presents the framework developed in Java in case of easy implementation comparator in applications. There you will find a basic description of the classes and methods. Chapter 5 presents the design and implementation aspects of integration Brighthouse engine with comparators framework as a solution of the optimization. Section 8 presents data used on tests and experience and the ability to easily compare the results. The last chapter is a summary and an indication of possible directions for further research and development.

2. Algorithm

The algorithm consists of three phases of action: the segmentation phase, granulation and identification. The first phase is responsible for acquiring objects for the processing from contour map of Poland, in form of a single image area (object).

As a result of properly performed this phase, we get a lot of objects for testing, which are input to the next phase. An example of contour map and objects obtained are shown in Figure no. 1

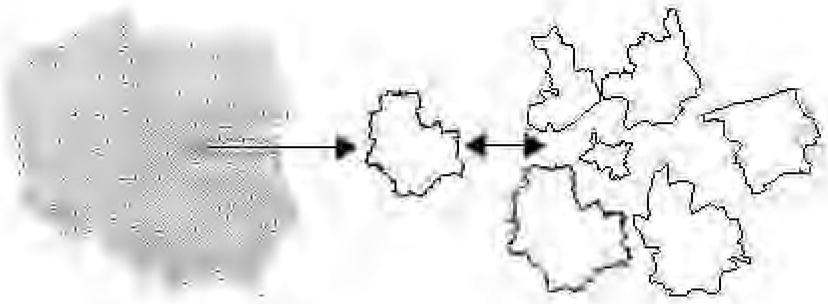
Figure 1: Example of input objects segmentation

The second phase is responsible for granulation of objects and save the input in the form of granules to allow the calculation of some characteristic values. The calculated values will be the determinant of a specific objects, which then will be used to identification. A detailed description was presented in a previous article [7]. Here, I limit the statement only to indicate that the algorithm provides for the calculation of the two determinants for each object, one is a "cover" a second "contour". Determinant of "coverage" counts the coverage rate of the granule by the geographical area. This is performed using the technique of histogram calculation.

The determinant of "contour" is created by using fuzzy linguistic variables and calculation of characteristic values of each granule. As a representation of the object we get a string encoded with symbols indicating the appropriate linguistic variables. Thus prepared objects are subjected to the final phase - identification.

This stage is implemented using a comparator in total, as defined in the articles [5] [6] and [9]. Identification consists of checking similarities to the identified reference objects. We do this by defining two comparators for each object to be identified. The first comparator is responsible for designating similar of feature "contour" and the other of feature "coverage". For "contour" we can calculate distance between strings (representing objects) using a Levenshtein metric and a fuzzy relations [1] [4] which was previously studied and described in the article [5]. The second comparator sets the similarity by checking of the differences in coverage percentage of granules [2] (for the test and reference objects) and compute the sum of differences. For both comparators we assume maximum function as a function of choosing the best solution for a given feature.

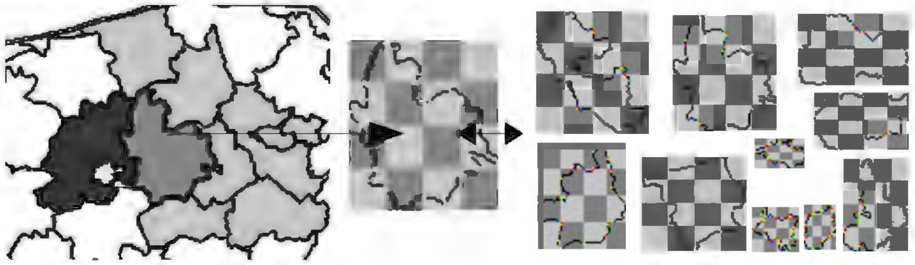
Figure 2: Granulation of objects and determination of characteristic points of the contour

3. Implementation of selected parts in phase I and II

In the realization of the first part we encounter some interesting implementation issues, i.e.: generation of one-pixel border of the area or building a coded contour by connecting the characteristic points of the border areas. In this article, we quote only fragments. If you are interested in please refer to [7], which describes in a comprehensive manner the various phases of the algorithm.

3.1 One-pixel border generation

The algorithm task is to create a border of the harvested area in the color RGB (0,0,0) with a thickness of one pixel. To accomplish this task we will use the 4-cohesive neighborhood [3], which will be analyzed for the presence of color of the inner area. If this neighborhood has a pixel with a given pixel color and the test point is not part of the area then the current pixel is marked with black (fixed color for the outline). A detailed activity diagram is shown in Figure 3.

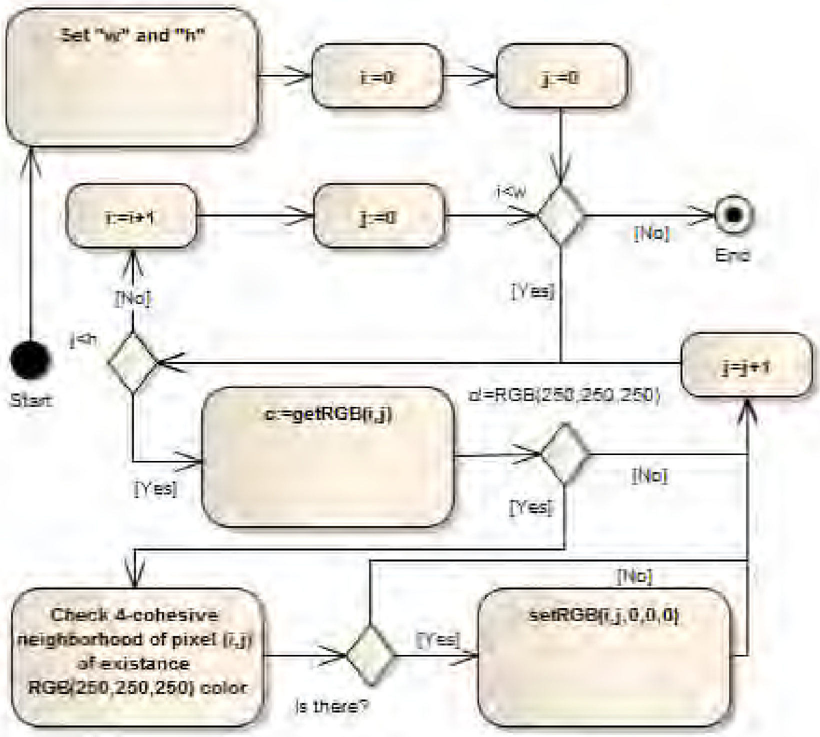

Figure 3: Activity diagram for one-pixel border generation

3.2 Border coding

The aim of the algorithm described above is to identify objects (contours) by the designation of the similarities and selection couples with maximum similarity. An important factor is the way of recording and comparing the contour. It is based on the simplistic approach of writing the outline and write transitions between some of its characteristic values along the border. In addition introduced fuzzification by the application of linguistic variables to describe the direction of the transition between characteristic points (assigned value for the different minimums and maximums of granules on the X and Y). The result is a sequence of encoded linguistic variables coded by the symbols R, L, U, D, RU, RD, LU, LD, where L is the shift on the left, R- right, U - up, D -down. It is worth noting that the coding does not distinguish between transitions at different angle crossing the line, but the same treat all calls that are inclined between 0 and 45 degrees, 45 and 90, etc. Thus, the deviations (noise) can be reduced.

The transition between the various reference points is along the border of the area. In order to optimize the transition steps from the starting point to the end (where the end point is also the start), it is implemented with memory "path" which is a remembering of visited pixels. This is to prevent deadlock of the algorithm and the minimization of visited pixels. Memory of visited pixels can retreat to the nearest pixel not visited in case you come across a "dead end". A part of the contour of the described situation is shown in Figure 4.

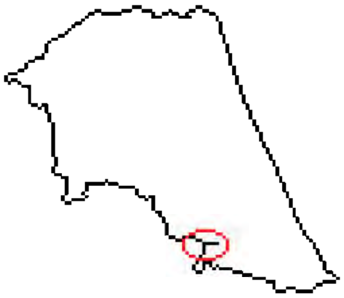

Figure 4: An example of a visible border with "dead end"

4. Comparator framework

The purpose to create a java library for handling compound objects comparators is the universalization of their use and to help solving similarity problem. Class Diagram [11] and methods is shown in Figure 5. The main advantage of this framework is able to apply to each type of object is not depending on the type as well as how to compare. The library consists mainly of abstract classes, which impose the need to implement some specific methods for specific types of objects. The result is that part of the universal and common set is implemented in those classes, and the specific part of object or its features, is implemented in the class of the specific object itself (dedicated Comparator). Table 1 shows the description of the purpose of classes and methods.

Table 1: Classes included in the framework

No	Class name	Abstract	Description
1	IdentifiedObjectAbstract	YES	Object class of the input objects that forces the implementation of feature representation method.
2	ReferenceObjectAbstract	YES	Reference object class, it forces the implementation of feature representation method for reference object.
3	ComputedObjectPair	NO	A class that represents a pair of objects and computed the resulting similarity between them.
4	CompoundComparatorAbstract	YES	Primary class of comparator implements the process of comparing and determining the value of similarity for a single feature. It forces implementation of method that calculates value of membership function to fuzzy relation.
5	DecisionComputer	NO	Class of aggregating results of individual comparators. Provides different methods of aggregation

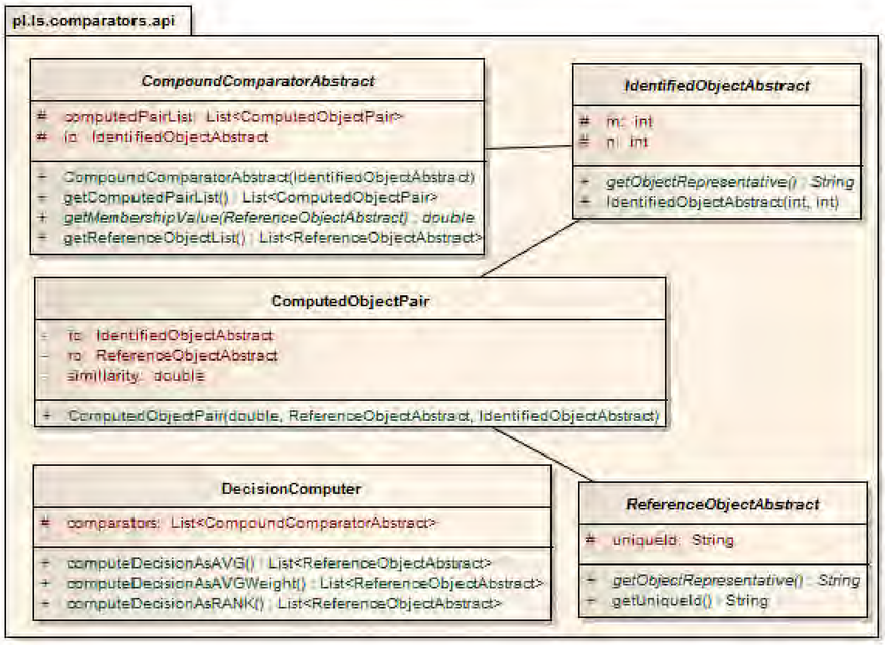
pl.ls.comparators.api
CompoundComparatorAbstract
# computedPairList : List<ComputedObjectPair>
# io : IdentifiedObjectAbstract
+ CompoundComparatorAbstract(IdentifiedObjectAbstract)
+ getComputedPairList() : List<ComputedObjectPair>
+ getMembershipValue(ReferenceObjectAbstract) : double
+ getReferenceObjectList() : List<ReferenceObjectAbstract>
IdentifiedObjectAbstract
# m : int
# n : int
+ getObjectRepresentative() : String
+ IdentifiedObjectAbstract(int, int)
ComputedObjectPair
- io : IdentifiedObjectAbstract
- ro : ReferenceObjectAbstract
- similarity : double
+ ComputedObjectPair(double, ReferenceObjectAbstract, IdentifiedObjectAbstract)
DecisionComputer
# comparators : List<CompoundComparatorAbstract>
+ computeDecisionAsAVG() : List<ReferenceObjectAbstract>
+ computeDecisionAsAVGWeight() : List<ReferenceObjectAbstract>
+ computeDecisionAsRANK() : List<ReferenceObjectAbstract>
ReferenceObjectAbstract
# uniqueId : String
+ getObjectRepresentative() : String
+ getUniqueId() : String

Figure 5: Comparators framework class diagram

5. ICE in implementation

In previous studies [9], ICE has been chosen as an environment for easy storage of data about objects in the form of ROLAP cubes [10] as a basis for data warehouse. For implementation of this algorithm we also can use this platform. This will enable easier access to data and faster to perform certain operations and checking to be performed using simple SQL queries without the need for highly expensive calculations on images.

Figures no. 6 and 7 show diagrams of ROLAP cubes which are designed using a star schema, used to store data about objects separated due to the studied features - "contour" and "coverage". Thanks to the cubes, the data are already converted to a suitably defined parameters granulation, which greatly accelerates the performance of the identification algorithm. These structures store data for the relevant definitions. If you want to identify object for the parameters for which the cubes do not have the data, then you must first fill cubes with calculated data. The advantage is single calculation complicated value for each object and their granules, and then reuse the data.

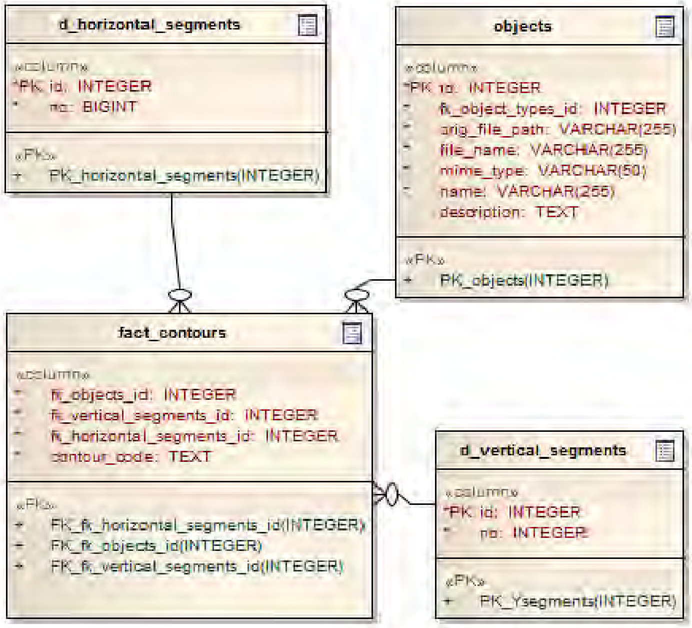

Figure 6: Schema of ROLAP cube that stores data of contours

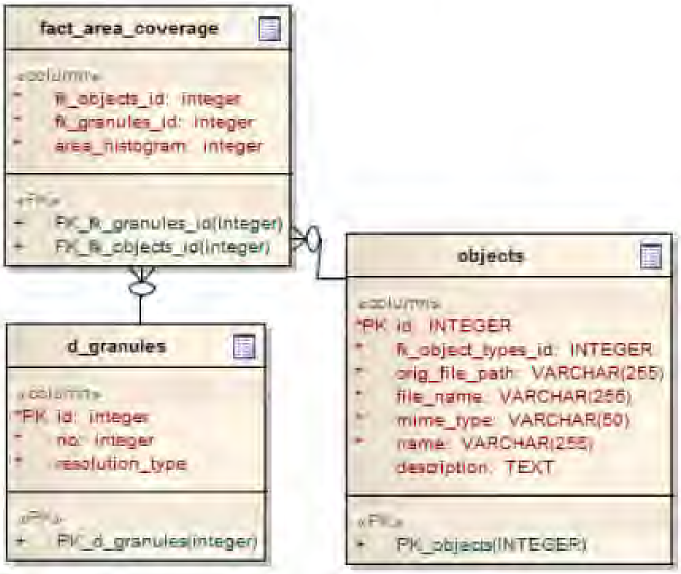

Figure 7: Schema of ROLAP cube that stores granules histograms used to calculate coverage

6. Method of selecting best results

The above method calculates the best solution for the features. The question remains, how to optimally use the results to choose the final solution? The issue is even more complicated that in general the algorithm can use more than two comparators (if you need to compare more features). Then each of the comparator returns the results of the test the similarity of features dedicated to. The results may be divergent. We must apply a solution that will choose the best solution considering the results of all the features. Here are some suggestions to solve this problem. In particular, each of these examples or also a completely different solution which is not listed here may be the best. It depends on the context and application.

6.1 The arithmetic mean

This method involves calculating the arithmetic mean of the similarities calculated by the different comparators for input object "a", relative to the same reference object. After calculating the average value of similarities, we choose the maximum value and the object for which the value has been obtained. Characteristic of this solution is prone to the dominance of one or more features in relation to other features. This is a level of similarity of the results returned by examining the feature comparator. If for one of features are returned high values of similarities (e. g. above 0.8) and for other characteristics, low (e. g., slightly above 0.5) then the first feature has a greater influence on the choice of solution, which is not always desirable.

6.2 Maximal rank

Another proposed method eliminates the above-mentioned problem, is to take into account the ranking of objects for the individual comparators. In this case, we are building numbering ranking positions in the ranking from 1 to n (for each comparator). Then we create sum of items for the same reference objects, but in different rankings. The final result we choose by selecting the minimum sum of items and the object associated with the sum. Special case is the situation where in all the rankings are the same reference object in the first place. Then this object is the result, without searching the entire space of sums rankings.

6.3 Weighted average

This variant is analogous to the case for the arithmetic mean. However, here we consciously give weights reflecting the importance of the attribute (feature). Please note that the weights should be chosen to adequately mitigate the problem raised for the arithmetic mean.

7. Results

The data used for experiments is contours map of Poland measuring 1050 x 1050 pixels with marked border of counties and cities with county rights. As a reference set we use county maps (single), which we have information about the code TERYT (GUS), which described objects. The results in Table 2 show the effectiveness of individual variants.

Table 2: Results of experiments for a set of counties and cities with county rights

No	**Name**	**Kk**	**Kp**	**AVG**	**RANK**	**AVGW**
1	Identification correctness %	54,90%	90,50%	92,00%	79,94%	91,02%
2	Time	33,9s	13,2s	40,0s	39,0s	41,0s
3	Number of incorrectly identified	171	36	30	76	34

Kk – identified only by examining the features contour

Kp – identified only by examining the features coverage

AVG – identification by both features using the arithmetic mean

RANK - identification by both features with rankings

ANGW - identification by both features using a weighted average with weights 0.75, 0.25 in favor of coverage

Space objects for identification consisted of 379 elements (the number of counties and cities with county rights in Poland). Reference set consisted of the same number of elements, and have all the appropriate objects to obtain correct results .

Tests were performed on a laptop with Centrino Duo processor, 8Gb RAM, 500GB 5400RPM drive.

Summary

In this paper, we find confirmation of the applicability of compound objects comparators to identify objects. We can find working examples used in large commercial projects. The text also allows for comparison of results, depending on the assumptions and used techniques.

Further possible activities are focused on methods of aggregation of comparators results as well as defining and testing of new other features (comparator) that can improve the quality and effectiveness of the method.

From another point of view, we should also check to what extent the presented method is general and suitable for other objects, less specific, or maybe even a different type (not images).

Bibliography

[1] Kacprzyk J. (2001), *Wieloetapowe sterowanie rozmyte.*

[2] Pedrycz W., Kreinovich V., and Skowron A. (eds) (2008),: *Handbook of Granular Computing*. Wiley

[3] Russ J. (2006), *The Image processing Handbook.*

[4] Rutkowski L. (2006), *Metody i techniki sztucznej inteligencji.*

[5] Sosnowski Ł. (2009), *Inteligentne dopasowanie danych przy użyciu teorii zbiorów rozmytych w systemach przetwarzania danych*. W: Analiza systemowa w finansach i zarządzaniu T.11 pod redakcją prof. J. Hołubca.

[6] Sosnowski Ł. (2010), *Budowa systemu porównywania obiektów złożonych*. W:Analiza systemowa w finansach i zarządzaniu T.12 pod redakcją prof. J. Hołubca.

[7] Sosnowski Ł. And Ślęzak D. (2011), *Comparators for Compound Object Identication*.13th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC-2011).

[8] Szczepaniak P. (2004), *Obliczenia inteligentne, szybkie przekształcenia i klasyfikatory.*

[9] Ślęzak and Sosnowski (2010), *SQL-Based Compound Object Comparators: A CaseStudy of Images Stored in ICE.* In: Proc. of ASEA, CCIS 117, Springer pp. 304-317.

[10] Todman C. (2005), *Projektowanie hurtowni danych.*

[11] Wrycza S. (2005), *Język UML 2.0 w modelowaniu systemów informatycznych*.

[12] Zadeh L. (1965), Fuzzy Sets. *Information and Control*, vol. 8

J. Hołubiec, red. –TECHNIKIINFORMACYJNE-TEORIAIZASTOSOWANIA,tom1(13)

ISBN 9788389475336