An Optimization Approach to Constraint-Based Generalization in a Commodity GIS Framework

Jean-Luc Monnot, Paul Hardy, & Dan Lee
ESRI, Redlands, California, USA
jlmonnot@esricartonet.com, phardy@esri.com, dlee@esri.com

ABSTRACT

The task of generalization of existing spatial data for cartographic production can be expressed as optimizing both the amount of information to be presented, and the legibility/usability of the final map, while conserving data accuracy, geographic characteristics, and aesthetical quality. This paper provides an overview of a research project underway presently at ESRI to implement an optimization approach to constraint-based generalization within a commodity GIS (ArcGIS). In this approach, a set of rules are defined, one for each constraint. Each rule contains a satisfaction function, measuring the degree of violation of the constraint, and one or more actions which should improve the situation if the constraint is violated. An Optimizer kernel then has the responsibility of evaluating local and global satisfaction, and applying actions to appropriate features to improve the situation. In real generalization scenarios, it is often not possible to avoid some violation of constraints, and the goal of the Optimizer is therefore to maximize the overall satisfaction.

This paper describes the concepts and components needed to achieve optimization, the mathematics of the optimization process, and outlines a research prototype implementation. It also covers mechanisms for conserving topological integrity, which are built into the optimization framework. It then describes a set of example use cases, particularly covering displacement, but also others such as contextual simplification.

Primary Conference Theme: 10 – Cartographic Generalization & Multiple Representations

1 INTRODUCTION

There are few commercial GIS products providing automated generalization tools, and most of those tools process a feature (or a feature class) at a time, applying a single generalization operation independent of context, and without considering other constraints that would impact the appropriate representation of the affected features. These tools are effective, but applying the initial operation can often expose further problems. Typical examples include simplifying a boundary, which may cause a nearby point feature to fall on the opposite side of the boundary; or displacing a building away from roads, which may move it over water.

Lack of context also means that two similar features in different parts of the map will always be treated the same, whereas for maximum clarity they should be processed differently (if one is in a rural area with lots of room, and another is in a dense urban area). In contrast, a human cartographer carrying out generalization will analyze the spatial context and decide which operators to apply to which feature in order to best preserve that context. The problems have been covered in a previous paper: “Geographic and Cartographic Contexts in Generalization” [Lee 2004]. To overcome these problems, we need to introduce the concepts of ‘Constraints’ and ‘Optimization’, and of an ‘Optimizer’ that applies them to geographic data.

2 CONCEPTS OF CONSTRAINTS AND OPTIMIZATION

2.1 Constraints

The concept of 'constraints' as a way of defining the requirements and goals of generalization has been actively researched for more than a decade [Beard 1991], and was explored comprehensively in a research summary [Ruas 1999]. Beard classifies constraints as: Graphical (e.g. minimum legible size), Structural (e.g. connectivity of roads), Application (e.g. importance of information content), or Procedural (e.g. transportation generalization comes after hydrography generalization).

Constraints were central to the design of the European AGENT project, which prototyped a multi-agent approach to constraint-based generalization [Lamy 1999]. Although powerful, the resultant multi-agent system introduced overheads of complexity and performance, and required an active object database infrastructure not readily available in a commodity GIS environment, thus limiting its applicability.

2.2 Optimization

The concept of mathematical optimization of a system by convergent evolution has an even longer pedigree, with key points being the Metropolis algorithm [Metropolis 1953], and 'simulated annealing' [Kirkpatrick 1983]. There have been various academic applications of simulated annealing to generalization, notably for displacement [Ware and Jones 1998].

Statistical optimization (such as simulated annealing) is a useful technique for finding a 'good enough' solution to the class of problems where determining an exact solution would require exploring a combinatorial explosion of possibilities. The classic example is the 'traveling salesman' problem – "Given a number of cities and the costs of traveling from any city to any other city, what is the cheapest round-trip route that visits each city exactly once and then returns to the starting city?" The most direct solution would be to try all the permutations (ordered combinations) and see which one is cheapest (using brute force search), but given that the number of permutations is n! (the factorial of the number of cities: n), this solution rapidly becomes impractical as n increases.

We assess through analysis of common use cases that geographic generalization (both model generalization and cartographic generalization) is in the same class of combinatorial problem, for which optimization is a good approach. This paper describes an Optimizer component, designed to apply optimization techniques to geographic data in a GIS.

Note however, that unlike previous applications of simulated annealing for generalization, the Optimizer has two significant advances:

- When a constraint is violated, the corresponding action is not a random response, as in many Monte-Carlo approaches. Instead, the action routine will apply the logic of generalization (using the spatial knowledge and neighborhood relationships of the GIS object toolkit) and make an intelligent change which is much more likely to result in improvement of overall system satisfaction.
- Although an action is triggered as a result of a constraint violation by a specific feature, the action routine may well modify other implicated features in order to improve the overall satisfaction. This mechanism helps minimize problems of cyclic behavior, and speeds convergence.

3 COMPONENTS

A set of basic concepts and components involved in an optimization solution have been defined; they are as follows:

3.1 Area (or set) of interest

An area or set of interest is a limited zone containing a number of relevant features where we want to solve an optimization problem (e.g. a block of buildings delimited by a set of roads in a cartographic generalization). This is the 'context space' for the generalization.

3.2 Action

An action is a basic algorithm, designed to improve satisfaction, with the following capabilities:

- Is invoked against a specific input feature.
- Can change that feature (or several features at a time).
- Declares the object classes it deals with and the parameters it needs.

Within the Optimizer system, an action is implemented as a dynamic COM object, which is linked via an XML definition to a constraint to make a rule.

3.3 Constraint

The process is led by constraints. A constraint:

- Provides a measure of satisfaction of a feature based on its environment (meaning that several other features may be involved in satisfaction calculation).
- Declares the object classes it deals with and the parameters it needs.

Within the Optimizer system, a constraint is implemented as a dynamic COM object, which is linked via an XML definition to one or more actions to make a rule. Each constraint provides a satisfaction function (see below).

3.4 Satisfaction function (SF)

The degree of satisfaction of a constraint will be a number greater than or equal to zero and smaller than or equal to one. We will call F_i the feature with id equal to i (this id defines the class id and the object id), and $S_c(F_i)$ the satisfaction of constraint c for feature F_i with:

$$0 \le S_c(F_i) \le 1$$

By convention $S_c(F_i) = 0$ will represent the case where the constraint is not satisfied at all and $S_c(F_i) = 1$ where the constraint is fully satisfied.

Any constraint must implement a Satisfaction Function and will normally provide a User Interface (UI) for the user to define the requirements and tune the satisfaction function using relevant parameter inputs. Here are examples of different curves of satisfaction functions (Fig.1).

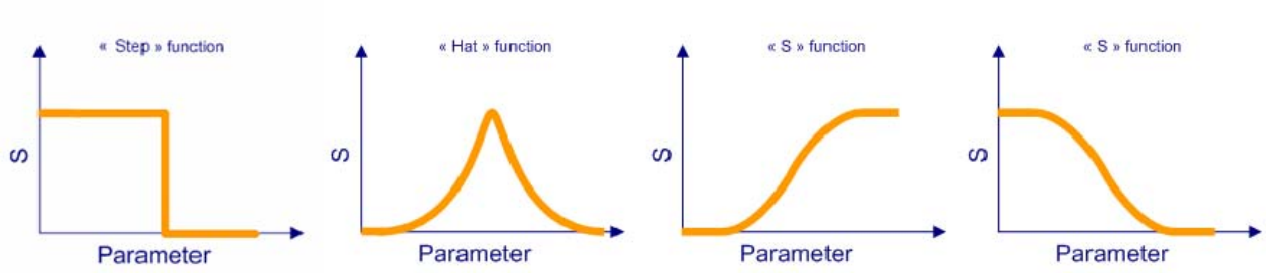

Fig. 1 – Satisfaction function curves

3.5 Optimizer kernel

The Optimizer kernel is the central component of the system. It is responsible for managing constraint satisfactions and executing related actions. The Optimizer:

- Applies a set of rules dedicated to the problem to be solved:
  - Set of constraints
  - Set of associated actions
- Focuses on one 'context space' (area/set of interest) at a time.
- Builds and provides all requested data for constraints and actions in the current context space.
- Caches frequently requested data in memory to optimize performance.
- Handles spatial structures such as topology and triangular neighborhood relationships.
- Manages the way actions are fired in order to reach the optimal state.
- Memorizes several modification sets and applies or aborts a modification set based on the increase or decrease of the global satisfaction.

The Optimizer kernel is implemented as a geoprocessing tool, which is linked via a geoprocessing model to one or more rules supplied as XML definitions which provide constraints and actions.

3.6 Rule

A rule is the association of one constraint with one or more actions. The meaning of a rule is:

"If constraint C is not respected then try action A1, then action A2 …"

3.7 Reflex

If implemented simplistically, the system would not respect some 'strict constraints' like "buildings MUST NOT overlap roads". This is because the Optimizer seeks for a balance between constraints to reach the best state.

Also, we anticipate the need for some data to be strongly linked to others. For instance the category for a building resulting from merging two initial buildings must be a function of the initial categories. This function is generally defined by a mapping organization in its product specifications.

The concept of a reflex is introduced to answer the two needs above. A reflex is a logical procedure fired after each data modification. It is responsible for filtering and modifying the results of the preceding action. A reflex can forbid certain system states (so can apply 'strict constraints') or it can propagate effects, such as by setting an attribute on the result feature.

3.8 Iteration

Having calculated the initial satisfaction for the set of features, the Optimizer has to choose one feature to become the target for the first iteration. This choice contains a random element, but is biased towards choosing a feature with a low feature satisfaction (tackle one of the worst problems first). For this feature, the constraint with the worst satisfaction will be chosen, and its actions tried, one by one. If the overall satisfaction improves, then the modifications are kept; otherwise, they are discarded.

A target feature for the next iteration is then chosen in a similar manner, and the process repeats. This continues until it reaches stability, or a maximum limit of iterations is reached, or other termination criteria are met (e.g. rate of improvement of satisfaction becomes negligible). Although it is fundamental that the choice of candidate for the next iteration has a random element, we can improve performance by taking advantage of the spatial nature of generalization to bias the selection towards taking nearer candidates first. Future research will investigate the benefits of introducing a systematic round-robin approach as an occasional alternative target selection mechanism to ensure that all features are visited at least once.

3.9 Temperature (simulated annealing)

In order to avoid being trapped by a local maximum we use the well known "simulated annealing" technique [Kirkpatrick 1983]. This strategy consists of accepting some action with negative ΔS, where ΔS is the difference between the current and previous satisfaction values. The algorithm is the following:

- Try actions and calculate best ΔS
  - if $\Delta S \geq 0$ then accept action modifications
  - else accept action modification with probability $\exp\left(\frac{\Delta S}{T}\right)$
- Decrease temperature T and continue iterations

The concept of temperature comes from analogy with annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one [Wikipedia 2006].

The decay rate α for temperature is one parameter of the Optimizer. The temperature function is exponential: $T^{t+1} = \alpha T^t$. For display convenience we choose to use a temperature starting with value 1 and decreasing towards 0.

3.10 Detection of cyclic behavior

One classic problem of dynamic systems like the Optimizer is that they can get locked into cycles of repeating states. Solutions to avoid this already exist, including the use of taking the Fourier transform of the overall satisfaction and looking for periodicity. We will also learn from the experience of earlier dynamic system approaches to generalization, such as the AGENT prototype.

3.11 Topology Cache

Topology is the branch of geometric mathematics concerned with order, contiguity, and relative position, rather than actual linear dimensions. As such, it is used to refer to the

continuity of spatial properties, such as connectivity or adjacency, which are unchanged after smooth distortion.

Topology is vitally important to good contextual generalization [Mackaness & Edwards 2002], but few if any existing systems are fully aware of topology during their generalization operations. Examples of generalization operations which can easily break topological relationships include simplification, elimination, aggregation, or displacement. The Optimizer kernel is designed round a geometry cache which is aware of topology. Another paper by the same authors [Monnot et al 2007] covers the relationship of topology to optimization generalization in much more detail, and will be presented at the ICA generalization workshop prior to the main conference.

4 SATISFACTION

Let N_f be the feature count. The $\langle\ \rangle$ brackets are used to symbolize the statistical mean operator.

The satisfaction of a constraint c for a feature F_i is defined by:

$$0 \le S_c(F_i) \le 1$$

The satisfaction of a constraint c for the whole system is:

$$S_c = \langle S_c(F_i) \rangle_i = \frac{1}{N_f} \sum_i S_c(F_i)$$

Satisfaction for a feature is:

$$S(F_i) = \langle S_c(F_i) \rangle_c = \frac{1}{\sum_c w_c} \times \sum_c w_c \; S_c(F_i)$$

where w_c is the weight applied to a constraint satisfaction; a larger weight makes one constraint more important than another.

The global satisfaction is the average satisfaction for all constraints and features:

$$S = \frac{1}{N_f} \times \frac{1}{\sum_c w_c} \times \sum_c \sum_i w_c \; S_c(F_i) = \langle S_c \rangle_c = \langle S(F_i) \rangle_i$$

The goal of the Optimizer is to maximize the global satisfaction.

4.1 Satisfaction calculation over iterations

As S is a linear operator of constraint satisfactions, it is easy to evaluate the difference in global satisfaction ΔS following any action modifications:

$$\Delta S = \frac{1}{N_f} \frac{1}{\sum_c w_c} \sum_c \sum_i w_c \; \Delta S_c(F_i)$$

Making the assumption that an action will not modify a huge set of features, we see that this quantity will involve few sums to be calculated. If ΔS is positive, then the modification is good and will be accepted. If ΔS is negative, then the modification may be accepted if the temperature is high (to pass through a worse state to get to an even better state), but otherwise, the modification will be backtracked and another action tried instead.

5 IMPLEMENTATION AND DEPLOYMENT

5.1 Implementation

The Optimizer and the rule-condition-constraint-action mechanisms are being prototyped within the geoprocessing environment of ArcGIS. This facilitates building the optimization stages into bigger process models, using the ModelBuilder framework. These models can automate the complete data derivation and production workflow, including data enrichment, partitioning, clustering, analysis and optimization, as well as more traditional uniform generalization (selection, classification, simplification, etc).

5.2 Optimizer XML Model

The full definition of the Optimization process is contained in an XML file which lists the components involved in the problem: Constraints, Actions and Reflexes. Each component is described by a section in the XML file as shown in the following example:

```
<[Constraint, Action, Reflex]>
        <Name>Display Name</Name>
        <ComponentName>Internal Component Name</ComponentName>
        <Parameter>
                <Name>Parameter Name 1 as Declared By the component</Name>
                <Value>[10, true, feature class…]</Value>
        </Parameter>
        <Parameter>
                <Name> Parameter Name 2 as Declared By the component </Name>
                <Value>…</Value>
        </Parameter>
</[Constraint, Action, Reflex]>
```

Parameter values may be defined in the `<Value>` tag or they may reference a value defined somewhere else in the XML file.

```
<ParameterValue>                                        Parameter value definition
        <Name>Simplification Tolerance</Name>
        <Type>Float</Type>
        <Value>30 meters</Value>
</ParameterValue>
…
<Constraint>…
        <Parameter>…
                <Value>byref Simplification Tolerance</Value>      Reference to value
        </Parameter>
</Constraint>
<Constraint>…
        <Parameter>…
                <Value>True</Value>                     Value without reference
        </Parameter>
</Constraint>
```

Optimizer Rules are also included in the XML model using the following pattern:

```
<Rule>
        <Name>Display Name</Name>
        <Constraint>Constraint Name</Constraint>
        <Action>Action 1 Name</Action>
        <Action>Action 2 Name</Action>
        <Action>…
        <Weight>1.5</Weight>
</Rule>
```

The Optimizer is exposed to the end user as a geoprocessing tool, within the ArcGIS geoprocessing framework. The visual ModelBuilder can be used to connect the Optimizer to rules (XML), or tool parameters can be entered directly, as shown in Fig. 2.

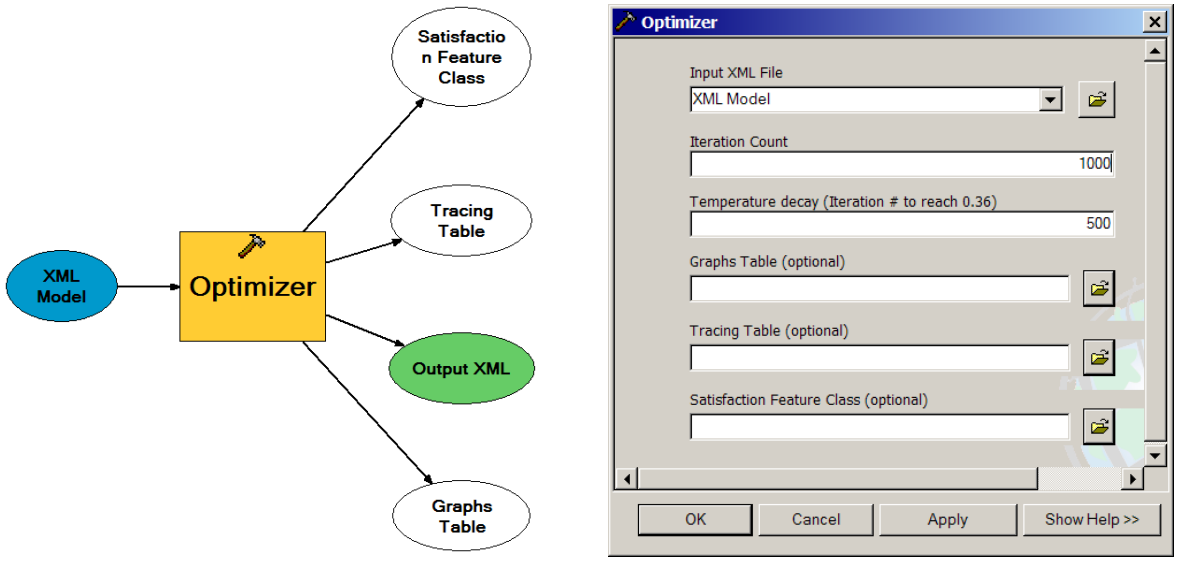

Fig. 2 – Optimizer in ArcGIS Geoprocessing model, with tool parameters

The outputs of the Optimizer include a set of tables that are useful for analysis, trace and debugging. These are described further in Annex 1.

5.3 Optimizer XML Template

We anticipate that the same Optimization strategy will often be reused with a different set of parameter values. For instance, a general “Geometry Simplification” may be reused to simplify different feature classes at different map scales. An XML template will be provided to facilitate the reuse of an existing strategy. A template is a model containing undefined parameter values. A geoprocessing tool is provided to transform a template into a model by setting parameter values. This tool provides a way to chain the Optimizer tool with any other tool (“Select” tool in Fig. 3 example).

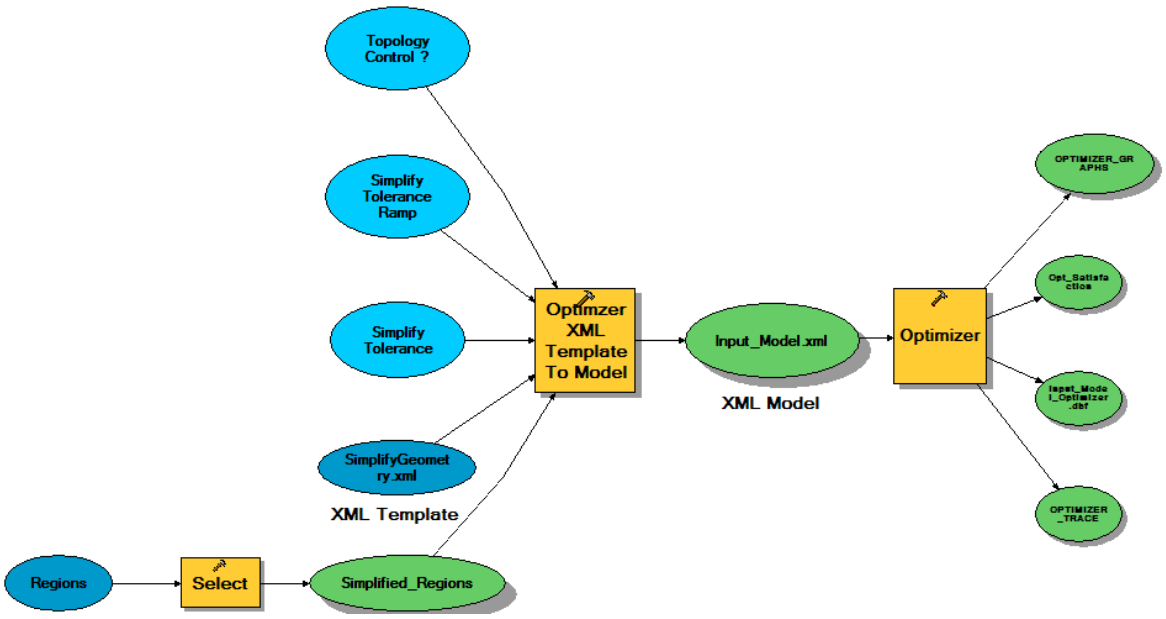

Fig. 3 - Deriving a Model from a Template and chaining Optimization with other tools

5.4 Deployment

Generalization forms part of the wider task of derived data and multi-scale map production, an increasingly mission-critical strategy for national and regional mapping agencies and commercial map and geodata publishers. Contextual generalization is the key component in enabling efficient generation of multiple products from a master database.

The generic nature of the Optimizer will also extend its applicability beyond traditional generalization, such as in the combinatorial aspects of cartographic representation. An example is laying out multiple bus routes which share a common centerline (as a bundle of offset lines) while maximizing continuity and minimizing crossings, and ensuring that start and end of routes lie at the outside of the route bundle. Similar optimization algorithms have been used in a prior related graphical offsetting implementation for Paris bus maps (Fig. 4a).

The Optimizer will also be applicable to non-cartographic problems within the GIS, such as assigning polygons to equivalence sets, subject to constraints of equality of size and minimal shared boundary. This type of requirement is common in situations where land is to be exchanged or traded in order to consolidate fragmented holdings (Fig. 4b).

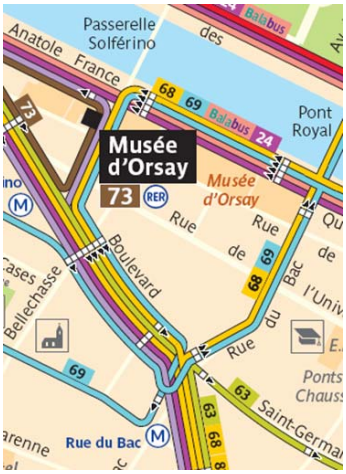

Fig. 4a – Bus route depiction (Copyright RATP, Paris)

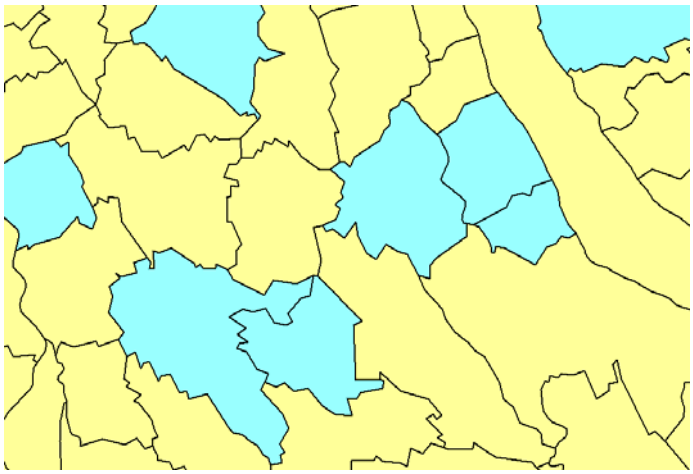

Fig. 4b – Assigning polygons to constrained sets

6 EXAMPLES

The Optimizer prototype is being tested with a variety of generalization scenarios. For the purposes of this paper, to explain the operation of the Optimizer, we will take a very simple point displacement scenario, involving two or three constraints.

Note that it is the cartographic representations that we want to displace, not the actual point features. ArcGIS 9.2 has introduced rich capabilities for storing and displaying rule-based cartographic representations with overrides, as described in the paper on "GIS-Based Generalization and Multiple Representation of Spatial Data" [Hardy 2005].

6.1 Example data

The initial data contains a range of geographic point features, being symbolized as graphical markers (colored circles in Fig. 5a). This kind of data occurs frequently in a range of vertical markets, such as the petroleum industry, forestry, and so on. It is obvious that the initial representation of this data is not good – the symbols overlap, and some may be hidden totally. We want to displace the representations of the features so that they do not overlap.

6.2 Example 1 – displacement using two constraints

The following two simple constraints are added:

- No Overlap – the symbols (treated as circles) should not overlap
- Small Offset – the symbols should not move far from their original positions

And the two corresponding simple actions are:

- Move away – displace away from contact
- Move towards – displace towards original position

The result of running the Optimizer with those constraints and actions is shown in Fig. 5b. The symbols have been displaced so as not to overlap, but without moving further than necessary from their original locations.

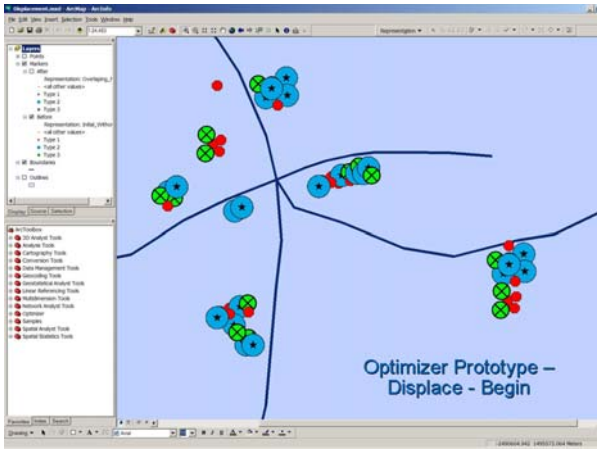

Fig. 5a – Initial state

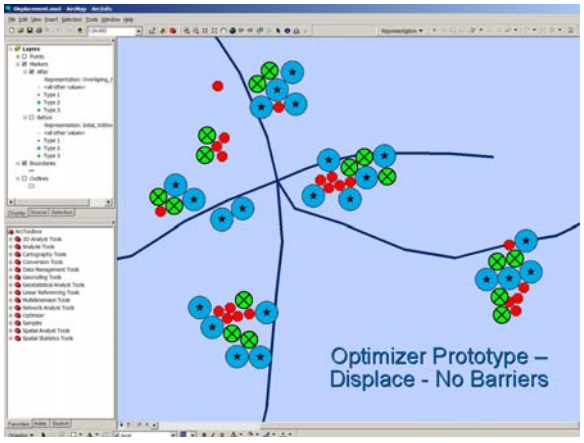

Fig. 5b – Optimized, without barriers

The evolution of the satisfaction for each constraint (and the overall satisfaction) is shown in Fig. 6a. Also shown is the decay of temperature as the iterations progress. The sudden step change in the satisfaction values in the right of the graph shows where the constraint satisfaction for the 'small offset' constraint has had to get worse in order that the satisfaction for 'no overlap' can get much better, resulting in an improvement in overall satisfaction.

6.3 Example 2 – displacement with added barrier constraint

Fig. 6b shows the results of introducing a third constraint, that the point symbols should not move to overlap the linear features (roads), which demonstrates the adaptability and extensibility of the system. This constraint is a 'strict constraint' or prohibition, and hence does not need an action, as it uses the 'Reflex' mechanism to forbid any state violating the constraint.

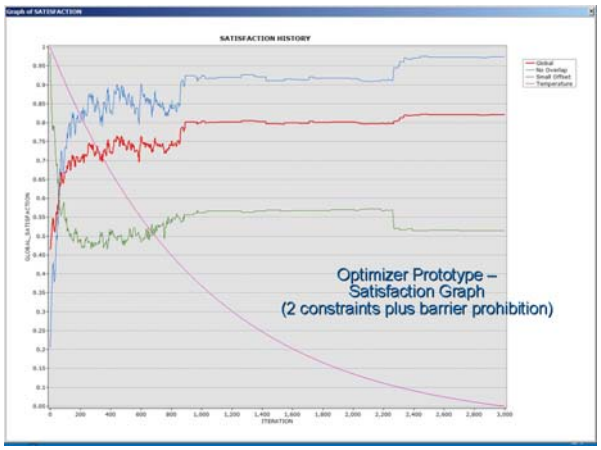

Fig. 6a – Satisfaction against temperature

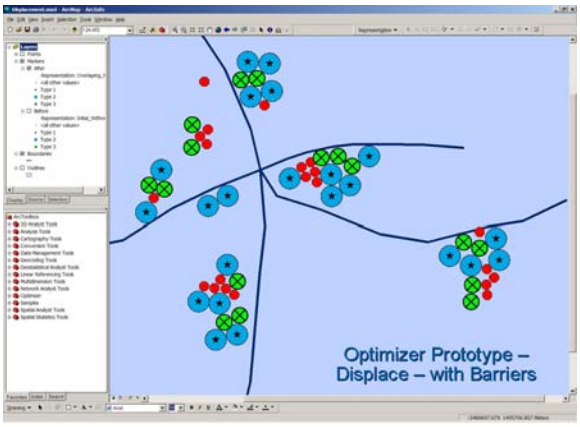

Fig. 6b – Optimized, with barrier constraint

6.4 Example 3 – geometry simplification using two constraints

The initial data is a polygon feature class containing features with complex shapes. The goal is to simplify those features by removing some details. Input geometries are decomposed by the Optimizer into a topological model containing nodes and segments with relationships between these two categories of objects. The objects the Optimization is evolving are segments. Two constraints are used to lead the simplification:

- Segment should be relevant
- Segment should be representative

The corresponding actions are:

- Merge segment with the longest linked segment (associated to the first constraint)
- Merge segment with the shortest linked segment (associated to the first constraint)
- Split segment (associated to the second constraint)

Initial shapes: 120,000 segments

Optimized shapes: 373 segments

Fig. 7 - Shape simplification obtained by Optimization

6.5 Example 4 – graphic displacement using two constraints

The input data is a layer displaying a point feature class. Each feature is represented by a marker symbol. Graphics overlap at some places. The goal is to move features to suppress graphic overlaps. In contrast to example 1 above, the true shape of the marker symbols is used to ensure a close packing of displaced symbols. The moving process is led by two constraints:

- Point should not be displaced too much
- Graphics should not overlap

We associate the following actions:

- Move back to original position (associated to the first constraint)
- Switch position with one neighbor (associated to the first constraint)
- Move away from overlap (associated to the second constraint)

The initial configuration and result is shown in Fig. 8:

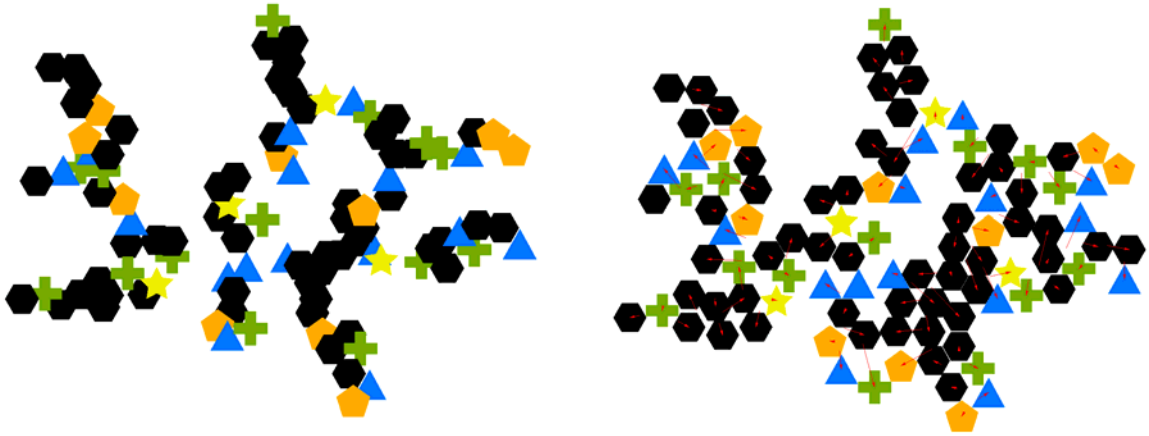

Fig. 8 - *Dispersing graphic markers by Optimization.*

7 FUTURE DIRECTIONS

The work on the Optimizer so far has been to prove the concepts and mechanisms, using simple constraints and actions, for typical generalization examples. This task is continuing as we learn from the experience gained.

The next task is to define a variety of use case scenarios, ranging from simple to complex, and covering different industries. Sample use cases will then be selected and implemented using the rule-condition-constraint-action paradigm described above. These use cases will involve multiple constraints and prioritizing of generalization actions, etc., so that the system can perform and demonstrate “comprehensive” generalization tasks.

Beyond that, a transition from prototype to product is dependent on a range of technical, integration, performance, market and commercial considerations. However, the current progress is very encouraging.

8 SUMMARY

- A design is proposed for an Optimizer approach to contextual generalization, for inclusion in a commodity GIS (ESRI ArcGIS).
- The design builds on decades of mainstream IT experience in optimization, but takes advantage of the spatially aware development environment of the GIS.
- The design has been prototyped and first results are very promising.
- The Optimizer design is generic and has applicability in GIS beyond generalization.

NOTE

This paper is a forward-looking research document, and the capabilities it describes are evolving prototypes. As such, it should not be interpreted as a commitment by ESRI to provide specific capabilities in future software releases.

9 REFERENCES

1. Lee D. 2004, “Geographic and Cartographic Contexts in Generalization”, ICA Workshop on Generalisation and Multiple Representation, Leicester, UK, August 2004. http://ica.ign.fr/Leicester/paper/Lee-v2-ICAWorkshop.pdf
2. Beard K. 1991, “Constraints on Rule Formation” in: Buttenfield, B.P. and McMaster, R.B. (eds.). Map Generalization: Making Rules for Knowledge Representation. London: Longman, 121-135.
3. Ruas, A. “Strategies de généralisation de données géographiques à base d’autonomie et de contraintes”, PhD thesis, Marne-La-Vallée, 1999 (in French).
4. Lamy et al 1999, “AGENT Project: Automated Generalisation New Technology”, 5th EC-GIS Workshop, Stresa, Italy, June 1999. http://agent.ign.fr/public/stresa.pdf
5. Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M.; Teller, A. H.; and Teller, E. "Equation of State Calculations by Fast Computing Machines." J. Chem. Phys. 21, 1953.
6. Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. "Optimization by Simulated Annealing." Science 220, 671-680, 1983.
7. Wikipedia entry on “Simulated Annealing”, http://en.wikipedia.org/wiki/Simulated_annealing, visited 27 February 2007.
8. Mackaness, W; Edwards, G, “The Importance of Modelling Pattern and Structure in Automated Map Generalisation”, Joint Workshop on Multi-Scale Representations of Spatial Data, Ottawa, July 2002.
9. Monnot, J-L; Hardy, P.; Lee D. 2007 “Topological Constraints, Actions and Reflexes, for Generalization by Optimization”, ICA Workshop on Generalisation and Multiple Representation, Moscow, August 2007.
10. Ware J.M. and C.B. Jones (1998) “Conflict Reduction in Map Generalisation Using Iterative Improvement'” GeoInformatica 2(4), 383-407.
11. Hardy, P.; Lee D. 2005, “GIS-Based Generalization and Multiple Representation of Spatial Data”, Proceedings, International CODATA Symposium on Generalization of Information, Berlin, Germany.

Annex 1 - Analysis, Debug and Trace

The Optimizer can deliver three output tables, which are useful for analysis:

- GRAPH table (Fig. 9): contains parameter evolution information collected during the Optimization process. The main purpose of this table is to provide data for the user to build graphs displaying the evolution of different parameters against iteration count. It helps the user to understand how the constraints are balanced, how fast the Optimization reaches an acceptable result, etc.
- TRACE table (Fig. 10): contains the full history of the Optimization. One row is added per accepted action that modifies the data. The future use of this table is to provide a debug mode allowing the user/developer to replay an Optimization sequence in order to gain a better understanding of any "unexpected" results. Histogram graphs can be derived from this table as shown in the following examples.
- SATISFACTION feature class table (Fig. 11): contains one feature per object (input feature or sub-object like nodes, segments or triangles) targeted by Constraints. This feature contains satisfaction values and a point shape. It is used to obtain a spatial display of the final satisfaction, allowing the user to quickly check if there is still work to do.

ITERATION	GLOBAL_SATISFACTION	Points_Should_Not_Be_Closer_Than_A_Given_Distance	Points_Should_Not_Be_Too_Much_Displaced	TEMPERATURE	BAD_ACTION_RATE	ACCEPTED_BAD_ACTION_RATE	ELLAPSED_TIME	FORTUNE_WHEEL_POPULATION
289	0.916044	0.892606	0.962919	0.824757	0.4375	0.809028	0.187	100
293	0.917287	0.894301	0.963258	0.82256	0.454861	0.805556	0.187	100
297	0.919722	0.897831	0.963505	0.82037	0.46875	0.798611	0.187	100
301	0.918327	0.895284	0.964412	0.818185	0.480903	0.805556	0.203	100
305	0.916973	0.893171	0.964577	0.816006	0.491319	0.8125	0.203	100
309	0.921559	0.900195	0.964288	0.813833	0.494792	0.819444	0.203	100
313	0.921328	0.899895	0.964194	0.811666	0.494792	0.826389	0.203	100
317	0.922499	0.901998	0.9635	0.809504	0.491319	0.822917	0.203	100
321	0.922439	0.901888	0.963542	0.807348	0.493056	0.833333	0.203	100
325	0.921235	0.899887	0.963931	0.805198	0.496528	0.84375	0.218	100

Iteration →← Satisfactions →← Simulated annealing info →← Debug info

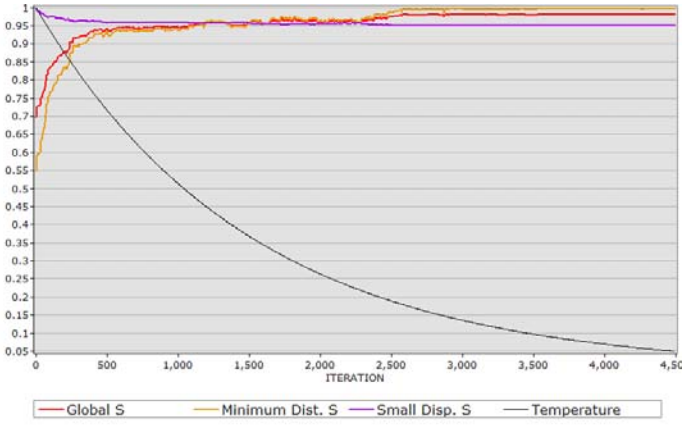

Fig. 9 - GRAPH table schema, content and display

ITERATION	CLASS	ROW	RULE	ACTION
1	Node_Class	3	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
2	Node_Class	55	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
4	Node_Class	9	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
5	Node_Class	26	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
6	Node_Class	78	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
8	Node_Class	93	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
9	Node_Class	85	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
10	Node_Class	5	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
11	Node_Class	68	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
12	Node_Class	28	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
13	Node_Class	26	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
14	Node_Class	62	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
15	Node_Class	22	Points Should Not Be Too Much Displaced	Move Point ALONE Back To Its Original Position
20	Node_Class	57	Points Should Not Be Too Much Displaced	Move Point WITH TOUCHING Back To Its Original Position
21	Node_Class	33	Points Should Not Be Too Much Displaced	Move Point ALONE Back To Its Original Position
22	Node_Class	57	Points Should Not Be Too Much Displaced	Move Point WITH TOUCHING Back To Its Original Position
23	Node_Class	48	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
28	Node_Class	26	Points Should Not Be Closer Than A Given Distance	Move Points Away From Overlap
30	Node_Class	34	Points Should Not Be Too Much Displaced	Move Point ALONE Back To Its Original Position

Iteration →← Object →← Non satisfied Constraint →← Applied Action →

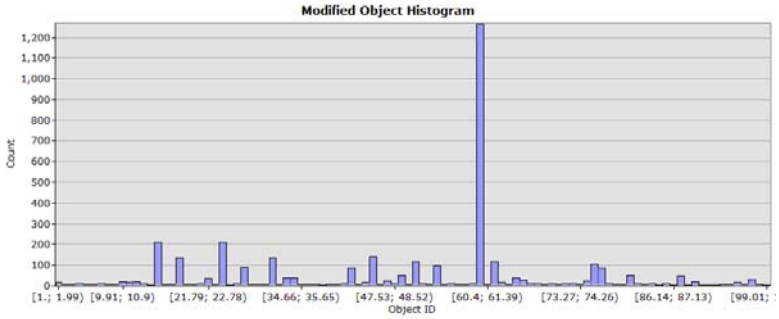

Fig. 10 - TRACE table schema and content. We build a histogram from the "ROW" column content to show that one feature is a real problem for the Optimizer.

CLASS	OBJ_ID	Points_Should_Not_Be_Closer_Than_A_Given_Distance_Class	Points_Should_Not_Be_Too_Much_Displaced_Class
Node_Clas	134	Good	Excellent
Node_Clas	135	Fair	Excellent
Node_Clas	136	Excellent	Good
Node_Clas	137	Poor	Excellent
Node_Clas	138	Fair	Good
Node_Clas	139	Excellent	Excellent
Node_Clas	140	Excellent	Excellent
Node_Clas	141	Excellent	Excellent
Node_Clas	142	Excellent	Excellent
Node_Clas	143	Fair	Excellent
Node_Clas	144	Excellent	Excellent
Node_Clas	145	Poor	Excellent
Node_Clas	146	Fair	Excellent

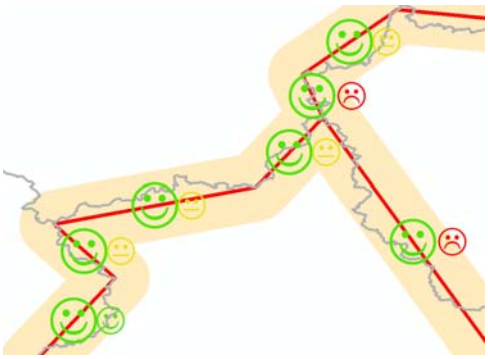

Fig. 11 - SATISFACTION feature class table and display. In this example, the segments are the objects on which the constraints are focused. Two constraints are involved in the Optimization. Large markers display satisfaction level for the first constraint and small markers for the second.

[Issue 1.0, of 2007-05-30]