# AN OPTIMIZATION APPROACH TO CONSTRAINT-BASED GENERALIZATION IN A COMMODITY GIS FRAMEWORK

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

**KEYWORDS**: Generalization, Optimization, Constraint, GIS, Contextual analysis

## 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 to maximize the overall satisfaction.

## 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 of an area 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 on "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 [Ruas 1999]. Beard classifies constraints as: Graphical (minimum legible size), Structural (connectivity of roads), Application (importance of information content), or Procedural (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 introduces overheads of complexity and performance, and requires 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.

We assess 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:

1. 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.
2. 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

The basic concepts and components involved in an optimization solution are as follows:

### 3.1 Area (or set) of interest

An area or set of interest is a limited zone containing a limited number of features where we want to solve an optimization problem (a block of buildings delimited by a set of roads in a cartographic generalization for instance). 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:

- Performs its task(s) based on an input feature. Can change several features at a time.
- Declares the object classes it deals with and the attributes it needs.

Within the Optimizer system, an action is implemented as a geoprocessing tool, which is linked via a geoprocessing model to a constraint to make a rule.

### 3.3 Constraint

The process is lead 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 attributes it needs.

Within the Optimizer system, a constraint is implemented as a geoprocessing tool, which is linked via a geoprocessing model 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).

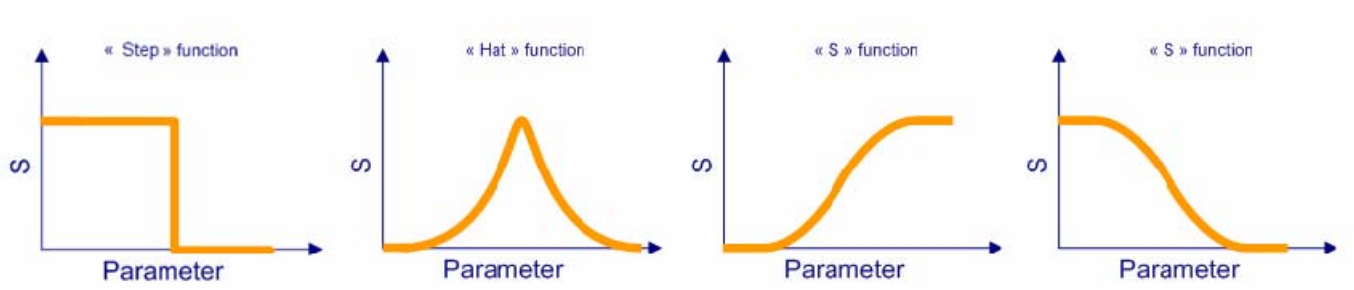


*Fig. 1 – Satisfaction function curves*

### 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.
- 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 which provide constraints and actions.

### 3.6 Rule

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

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

### 3.7 Condition

Generally a constraint will be defined at the feature class level ("buildings should not overlap"). But sometimes we will need to focus on a subset of features coming from the same feature class ("if building is a city hall it should not be merged with another one"). In order to address this requirement we introduce the concept of a 'condition' predicate, resulting in a true/false value P for a particular feature. This leads to the following kind of rule:

"If condition P is true, then if constraint C is not respected then try action A1, then action A2 …"

A condition will always be associated with a constraint, but a constraint may or may not have a condition.

### 3.8 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 the right 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 is a function of the initial categories. This function is generally defined by the particular organization's product specifications.

The concept of a reflex is introduced to answer the two above needs. A reflex is a piece of code fired after each data modification. It will be responsible for filtering and modifying the results of the preceding action.

### 3.9 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 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, else 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 satisfaction gain is sufficiently slow that iteration should stop. 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.

### 3.10 Temperature (simulated annealing)

In order to avoid being trapped by a local maximum we use the well known "simulated annealing" technique. This strategy consists in accepting some action with negative $\Delta S$, where $\Delta S$ is the difference between the current and previous satisfaction values. The algorithm is the following:

- Try actions and calculate best $\Delta S$
  - 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 by 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 $\alpha$ for temperature is one parameter of the Optimizer. The shape of this decay is exponential: $T^{t+1} = \alpha T^t$. For display convenience we choose to use a temperature starting with value 1 and decreasing toward 0.

### 3.11 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 are however known, including 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.

## 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:

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

$w_c$ are weights applied to each constraint satisfaction; they allow making one constraint more important than another.

The global satisfaction of the system 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 Optimizer goal is to increase 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 $\Delta S$ 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. The only remaining challenge is to be able to detect quickly which are the modified features and what is the impact on local satisfactions. This will be managed by modification sessions.

$S$ is a function of iteration count (pseudo-time). We will call $S^t$ the satisfaction at iteration $t$. When we apply the modification of a given action we have:

$$S^{t+1} = S^t + \Delta S$$

## 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 derived data production workflow, including data enrichment, partitioning, clustering, analysis and optimization, as well as more traditional uniform generalization (selection, classification, simplification, etc).

### 5.2 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. Optimization algorithms have been used in a prior related implementation for Paris bus maps (Fig. 2).

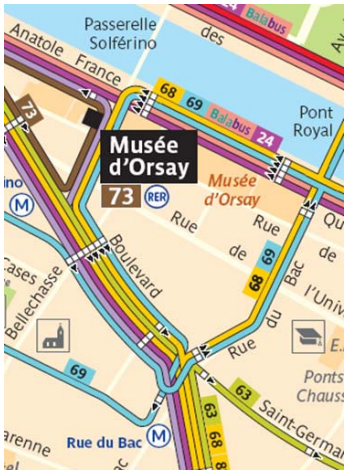


*Fig. 2 – Bus route depiction (Copyright RATP, Paris)*

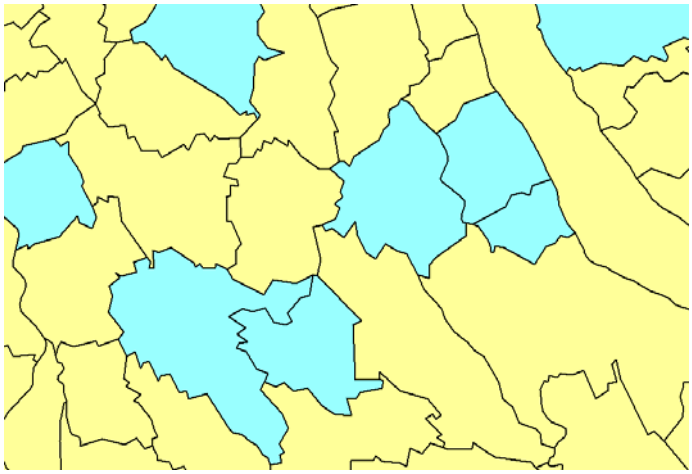

*Fig. 3 – Assigning polygons to constrained sets*

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. 3).

## 6 EXAMPLES

The Optimizer prototype is being tested with a variety of generalization scenarios, but 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, and not the point features themselves. 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 feature data, being symbolized as graphical symbols (colored circles in Fig. 4). This kind of data occurs frequently in a range of vertical markets, such as the petroleum industry, forestry, etc. 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 symbols so that they do not overlap.

### 6.2 Example 1 – displacement using two constraints

The following two simple constraints are added:

1. No Overlap – the symbols should not overlap
2. Small Offset – the symbols should not move far from original position

And the two corresponding simple actions are:

1. Move away – displace away from contact
2. Move towards – displace towards original position

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

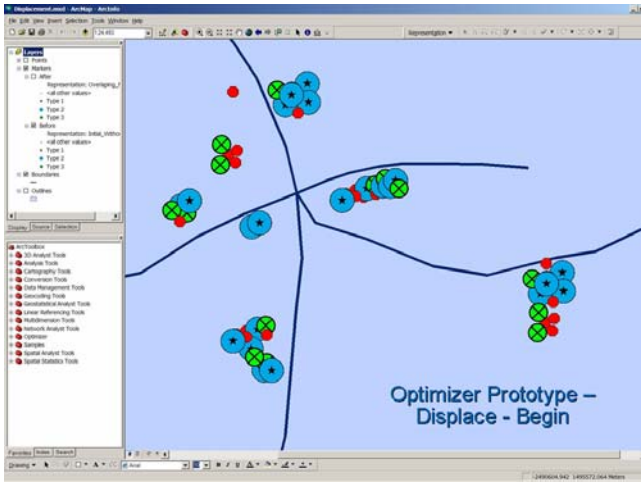


*Fig. 4 – Initial state*

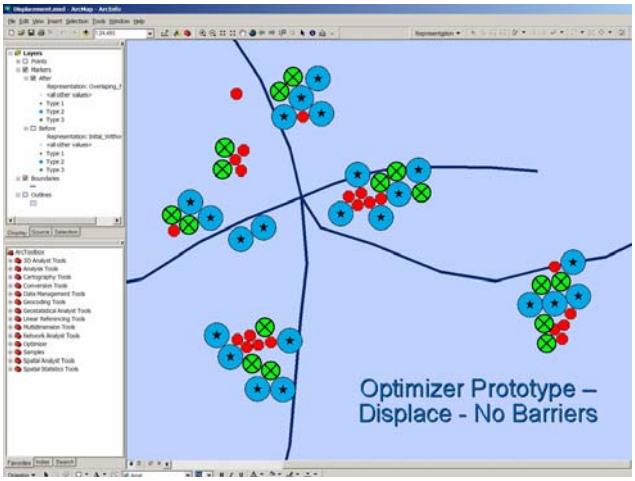


*Fig. 5 – Optimized, without barriers*

The evolution of the satisfaction for each constraint (and the overall satisfaction) is shown in Fig. 6. 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

In Fig. 7, we see the results of introducing a third constraint, that the point symbols should not move to overlap the linear features (roads), which shows 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.

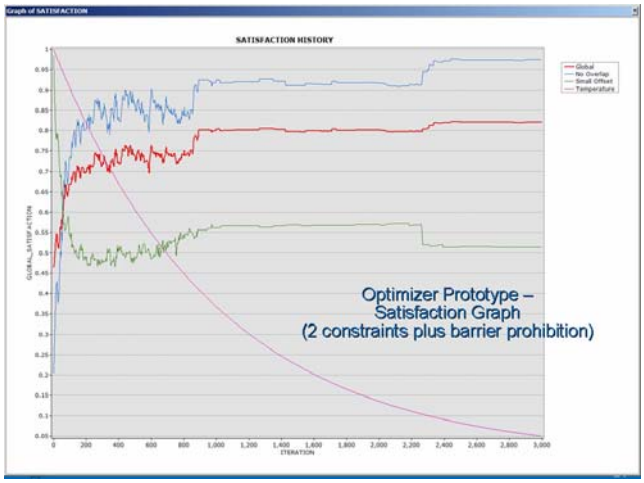


*Fig. 6 – Satisfaction evolution against temperature*

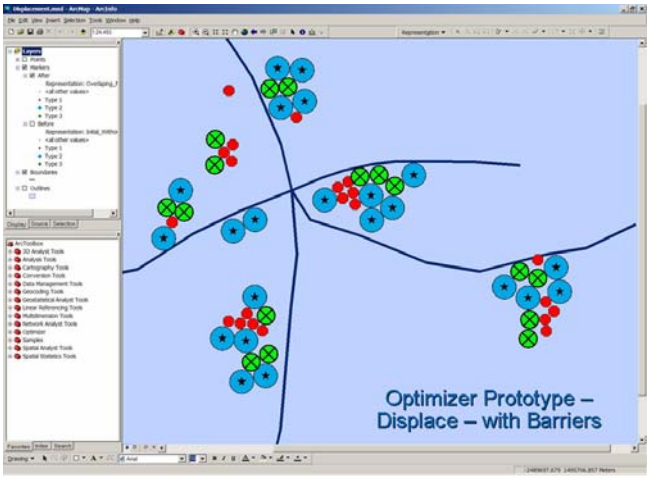


*Fig. 7 – Optimized, with barrier constraint*

The geoprocessing model used to execute the optimization is shown in Fig. 8. Note that all the Constraints, Actions, and Rules are all created by geoprocessing tools within the model. They, together with the data to be processed, in turn feed in to the Optimizer itself, which is also a geoprocessing tool.

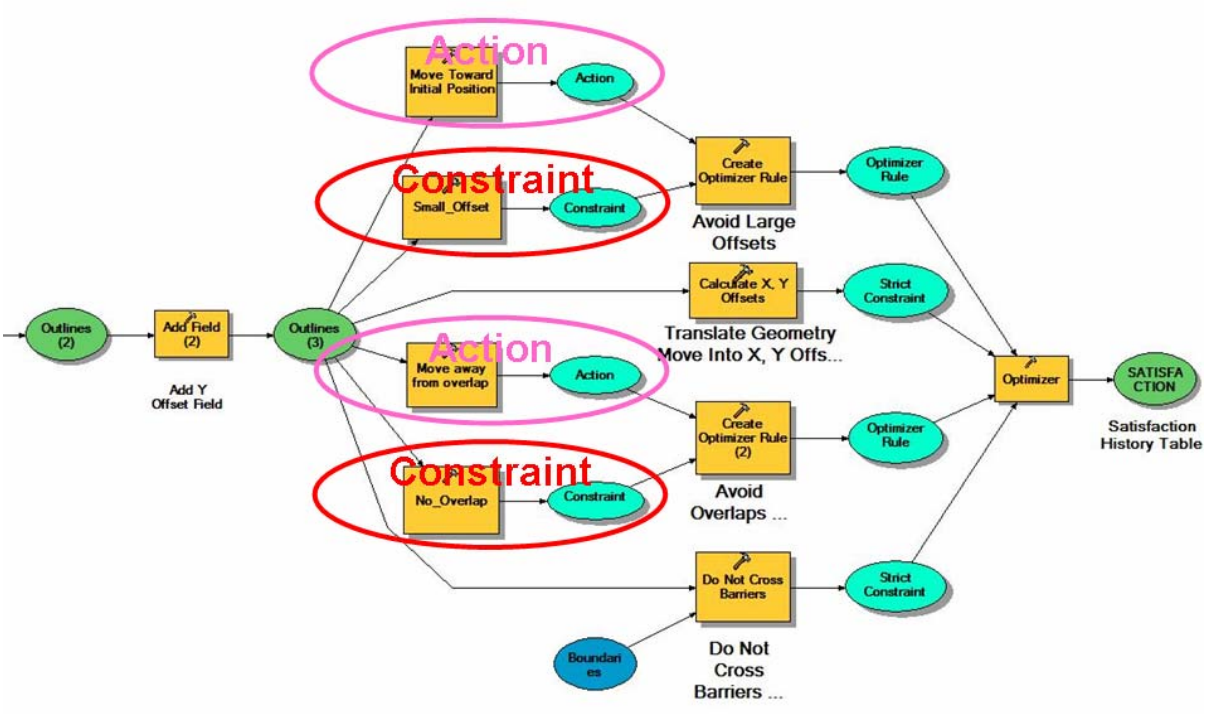


*Fig. 8 – Geoprocessing model for optimization*

## 7 FUTURE DIRECTIONS

The work on the Optimizer so far has been to prove the concepts and mechanisms, using simple constraints and actions, but 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, from the simple to the 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 would have applicability in the 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. 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.
2. Ruas, A. “Strategies de généralisation de données géographiques à base d’autonomie et de contraintes”, PhD thesis, Marne-La-Vallée, 1999 (in French)
3. 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
4. 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, 1087-1092, 1953.
5. Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. "Optimization by Simulated Annealing." Science 220, 671-680, 1983.
6. Ware J.M. and C.B. Jones (1998) “Conflict Reduction in Map Generalisation Using Iterative Improvement'” GeoInformatica 2(4), 383-407.
7. 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
8. Hardy, P.; Lee D. 2005, “GIS-Based Generalization and Multiple Representation of Spatial Data”, Proceedings, International CODATA Symposium on Generalization of Information, Berlin, Germany.
9. Wikipedia entry on “Simulated Annealing”, http://en.wikipedia.org/wiki/Simulated_annealing, visited 6 June 2006.


*[Issue 4, of 2006-06-08]*