User Hints for Map Labelling

Hugo A. D. do Nascimento[†,*]**, and Peter Eades**[*]
School of Information Technologies
University of Sydney
Sydney, NSW 2006, Australia
[hadn,peter]@it.usyd.edu.au

Abstract

The *Map Labelling Problem* appears in several applications, mainly in Cartography. Although much research on this problem has been done, it is interesting to note that map-labelling processes in commercial fields are very often executed manually or with little support of automatic tools. In general, practical map-labelling problems involve human subjective domain knowledge about label placement that is not entirely covered by the existing scientific approaches. In the present paper, we describe an interactive framework for minimising this technological gap. The framework allows users to interact with a labelling process by giving *hints*, which represent domain knowledge or adjustments that help an optimisation method to produce good labelling results. A map labelling system based on the framework has been implemented and an initial evaluation shows that it is promising.

Keywords: Map Labelling, user hints, combinatorial optimisation.

1 Motivation

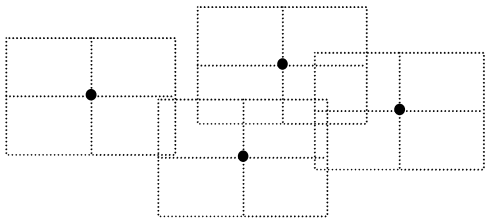
The general *Map Labelling Problem* consists of defining positions for the labels of several graphical features (cities, roads, lakes, rives, national parks, states, countries, etc.) of a map, such that these features can be uniquely identified. This is a well-known problem in Cartography that had its scientific foundation established with the research of Imhof (1962, 1975) and Yoeli (1972). Imhof and Yoeli defined a set of fundamental rules for labelling maps that has been used until the present time. These rules are:

1. **readability:** labels must have legible sizes;
2. **unambiguity:** each label must be easily identified with exactly one graphical feature;
3. **avoidance of overlaps:** labels should not overlap with other labels or other graphical features.

Since the debut of Map Labelling as a scientific area, this problem has increased in importance and has been subject of a considerable amount of publications. An illustrative chart of the distribution of the Map Labelling publications over the last 45 years can be seen in the *Map-Labeling Bibliography Web Site* (Wolff 1996).

Current map-labelling research has application not only in Cartography, but also in many other fields from the visualization of social networks to medical image analysis.

Techniques for solving Map Labelling problems in general adopt an abstract model about how to place labels. The most popular model is the *fixed position model*, where a finite set of candidate label positions is created for every feature (Neyer 2001). The labelling activity is then treated as a combinatorial optimisation problem which aim is to choose a subset of positions among the candidates, such that all features are labelled, and label-label and label-feature overlaps are avoided. Picture 1 shows a labelling example using the fixed position model for labelling four point features representing cities. For each city, four candidate positions are defined around it. The candidate label positions are represented by dotted rectangles. Note that some candidate positions overlap with each other. The labelling task aims to choose only one candidate position for every feature such that when placing the label on that position no overlap occurs. It is also possible to define preferences for the candidate positions in order to have labels placed on a particular side of their features as much as possible (e.g: on the top-right corner of the features). Preferences can be modelled as costs assigned to the candidate positions. The optimisation problem is then redefined as computing an overlap-free choice of label positions with minimum cost. In cases where there is no choice that results in an overlap-free labelling, then one or more features remain unlabeled.



Picture 1: candidate label positions for four cities. Each city is represented by a dot and has four candidate positions for label placement (represented by dotted rectangles around the feature).

The combinatorial optimisation problem stated above is NP-hard for most of its practical cases (Marks and Shieber 1991). Several heuristic techniques have been presented for Map Labelling, including Simulated

---

[†] Lecturer of the *Instituto de Infomática*, UFG-Brazil; Ph.D. scholarship from CAPES-Brazil.

[*] Supported by the Australian Research Council.

Annealing (Christensen, Marks, and Shieber 1994, 1995), Gradient Descent (Christensen, Marks, and Shieber 1995) and Integer Programming (Zoraster 1986, 1990, and Strijk, Verweij, and Aardal 2000).

Even though an extensive academic research on Map Labelling is available in the literature (Wolff 1996), it is interesting to note that these scientific results have not yet been applied broadly in industry. For instance, the labelling task performed by the main Australian companies that produce maps and street directories are still done manually or with little support of automatic map labelling tools (Russell 2002, Tollis 2002). In the international context, this problem is a issue of concern: the Computational Geometry Impact Task Force (Chazelle *et al.* 1999) has classified the Map Labelling as an important problem that needs to be investigated by future research.

We believe that the main reason for the outcomes of scientific research not to be applied in full for practical map labelling problems is the existence of a gap between the theoretical models studied in academy and real-life applications. Basically, real-life map labelling depends on *domain knowledge* about subjective aesthetical criteria and cognitive aspects of human vision that are not considered by many of the theoretical models. The existing optimisation methods are suitable for some particular map labelling aspects, mainly the avoidance of overlaps, as proposed by Imhof (1962, 1975) and Yoeli (1972), but very often they neglect other important issues such as ambiguity and "good visual appearance"[1] of the map.

In the present paper, we investigate how cartographers can interact with automatic map labelling techniques in order to convey domain knowledge to the process. We study a restricted case where only point features are considered, labels are modelled by rectangles of unit height, and there is a fixed number of positions where a label can be placed around its feature. We introduce an interactive framework that allows users to specify hints that help to produce good labelling solutions. A conflict graph of candidate label positions is used for modelling most of the user interactions.

The remainder of this paper is organized as follows: Section 2 presents a brief survey on interactive optimisation techniques. Section 3 introduces our interactive framework for map labelling. We discuss different ways of the user to interact with a labelling process and how these interactions can be implemented. Section 4 describes preliminary results obtained with the implementation of a map labelling system based on the framework. We show some labelling problems that can be alleviated by user interaction. Section 5 concludes with an overview of our approach and with topics for future investigation.

[1] When we say "visual appearance", we mean aesthetic aspects that make a labelling solution pleasant for the human eyes. Often, such aspects are subjective and intuitive for the users.

2 Interactive Optimisation

In recent years, there has been an increasing interest in the use of interactive tools for optimisation methods. This is partially due to developments in User Interfaces, more generally in the area of Human-Computer Interaction, which now can provide intuitive and effective environments for interaction with a variety of applications. On the other hand, there has been the realisation that several real-life problems still need humans as their main solvers. Examples are Garment industry bin packing (Gradišar, Jesenko, and Resinovič 1997), steel, wood and glass cutting (Beasley 1985), and vehicle routing (Anderson *et al.* 2000). There are a number of reasons for such need:

- the problem may depend on domain knowledge, including solution qualities, that are subjective for the user; this kind of knowledge may be difficult to describe precisely;
- the optimisation process may be dynamic, in the sense that some objectives and constraints are unknown prior starting the process;
- optimisation methods available for solving the problem may not handle all objectives and constraints, but only a small subset of these;
- computational resources (CPU power and memory size) may not be sufficient for solving the problem due to its complexity or to a large amount of data that has to be manipulated.

User interaction is necessary in this context in order to refine the problem or to manage the computational resources available. There are also well-known differences between human and machine skills for problem-solving, which can be exploited by interaction. Computers are suitable for intensive computation, where solutions for a problem are created and numerically evaluated; humans, on the other hand, are good in identifying patterns that differentiate good from bad solutions. Most heuristic optimisation algorithms (for NP-hard problems) easily find local optima (i.e., solutions for which there is no local improvement) but have difficulty finding global optima (solutions which are better than all other solutions). With good visual representations, humans can recognise that while a local improvement may be impossible, a more global improvement may be available.

In general, most real-life optimisation problems are handled in two steps: first an automated tool is applied, and then a person manually improves the solution in a post-processing stage. We are, however, interested in a more intense relation between the roles played by humans and automatic tools. We consider automatic tools as a natural extension of the human capabilities that should be used not only in the initial stage of an optimisation task but through the whole process.

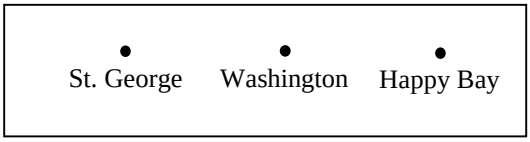
A good example of use of human interaction for optimisation problems is the cooperative paradigm called *Human-Guided Simple Search* (or *HuGSS* for short) presented by Anderson *et al.* (Anderson *et al.* 2000).

HuGSS divides an optimisation process into two main subtasks carried out by different entities: the computer is responsible for finding local minima using a simple hill-climb search, while the user works on escaping from local minima and leading the search to better solutions. The paradigm was successfully applied to the capacitated-vehicle-routing problem with time windows. A system was developed where the user could either change an existing solution to the problem manually, or focus a hill-climbing method to improve a particular area of the solution. A visualisation of the solution being improved provides feedback to the user and helps to decide the next action in the optimisation task. Experiments with the system showed that the performance of a simple hill-climb algorithm could be significantly enhanced by user interaction. Further investigations of the *HuGSS* paradigm were done (Lesh, Marks, and Patrigname 2000, Scott, Lesh, and Klau 2002).

Another example of interactive optimisation can be found in the area of Evolutionary Algorithms, where systems were developed to produce drawings of graphs by learning user preferences for a set of aesthetic criteria. The aesthetic criteria are highly subjective and manually setting their weights in a multi-objective function is a very time-consuming task (Mendonça 1994). The solution presented by Rosete-Suarez *et al.* (1999), Jacobsen (2001) and Barbosa *et al.* (2001) to this problem was to combine user-evaluation with an automatic optimisation method. A genetic algorithm produces a population of drawings, which are presented to the user. The user scores or ranks the drawings and the system uses this information to adjust the weights of the aesthetic criteria. The process repeats until the genetic algorithm generates a drawing whose quality reaches the user's expectation. One advantage of using genetic algorithms for this type of task is that they naturally produce several alternative solutions for evaluation. Moreover, genetic algorithms are flexible in handling a considerable amount of different objectives and constraints. A survey on Evolutionary Algorithms for interactive optimisation is presented by Takagi (2001).

We can say that both interactive approaches described previously use human interaction as a tool to fill in a "gap" (or weakness) in an optimisation process. The approaches differ mainly in the type of gap being considered. In the interactive graph drawing systems, the gap is an objective function that is not clearly defined *a priori*. The task performed by the user aims to clarify this function by providing indication about its parameters. In the vehicle-routing system, the objective function is well defined, but the optimisation method merely seeks local optima and does not find sufficiently good solutions. The user's role in this case is to guide the method to a global minimum through a better control of the optimisation method and the solution being developed. In the present paper, we cover both gaps: the user refines the map-labelling problem by inserting domain knowledge, and helps the optimisation method to converge to high quality labelling.

Map Labelling is a problem that can benefit from interactive optimisation. In fact, it already depends strongly on humans for evaluating labelling solutions. See, for example, the case in Picture 2 where three features are labelled as "St. George", "Washington " and "Happy Bay" respectively. Considered in this order the labelling may cause the user to misread the labels to say "George Washington" or "George Washington Happy". This can occur even though there is sufficient space in between the labels. The human map labeller may prefer to minimise this effect by positioning the label "Washington" above its feature; however, it would be very difficult to encode this problem as a general constraint in an optimisation tool.



Picture 2: ambiguity in labelling features.

Optimisation methods for map labelling commonly do not include such types of domain knowledge when computing a labelling solution. The user has to adjust the solution in a post-processing stage by manually moving some labels to different positions. Unfortunately, manual adjustment can also be a complex process, particularly when the map has several features close to each other, and the improvement of a single aesthetic issue causes new problems with the remaining labels. Therefore, a better way of combining inclusion of domain knowledge with automatic labelling optimisation is necessary. The ultimate goal is still to refine the problem, but minimizing now the amount of work that the user has to do.

In the next section, we describe an interactive approach for Map Labelling. It is based on our ***user-hints framework*** (Nascimento *et al.* 2001a, 2001b and 2001c) that aggregates several interaction facilities. The framework consists of a user, an optimisation method and a visualisation tool. The user gives *hints* to the optimisation method. *Hints* are adjustments aiming to refine the problem, reduce the space of solution to be explored, and escape from local minima. We consider three main types of hints:

- **Adjustments of objectives and constraints:** the user can change the objective function as well as add new constraints or remove existing ones. This is useful for inserting domain knowledge into the problem after starting the optimisation process.
- **Control of the optimisation method:** the user can focus the optimisation methods on parts of the problem. The idea is to concentrate action on areas that need major improvement. Focus is implemented by allowing the user to select and explore a subset of variables of the problem while fixing the remaining parameters. The complexity of solving a sub-problem is in general smaller than of the whole problem. As a result, much processing effort can be saved. The user can also control other aspects of the optimisation such as choosing a different

optimisation method or adjusting some tunning parameters of the method.

- **Manual changes:** all optimisation aspects that are not covered by constraints and focus can be managed by manual changes. In this case, the user directly adjusts the values of some variables of the problem. Such operation is useful when the user visualizes changes of a pre-computed solution that may lead to great improvement in quality.

These types of hints must be given in an intuitive way, preferentially through a graphical interface. The visualisation tool is a critical element of the framework, since it must supplies the users with feedback. Based on the visualisation, the user identifies quality aspects of the solution and decides what action to perform next.

We adopt a *supervised* model where the user runs the optimisation method when necessary. However, the framework automatically computes the quality of every new solution created by either the user or the method, and flags any improvement made on the best solution computed so far.

3 An Interactive Map Labelling Approach

An interactive system for Map Labelling should provide:

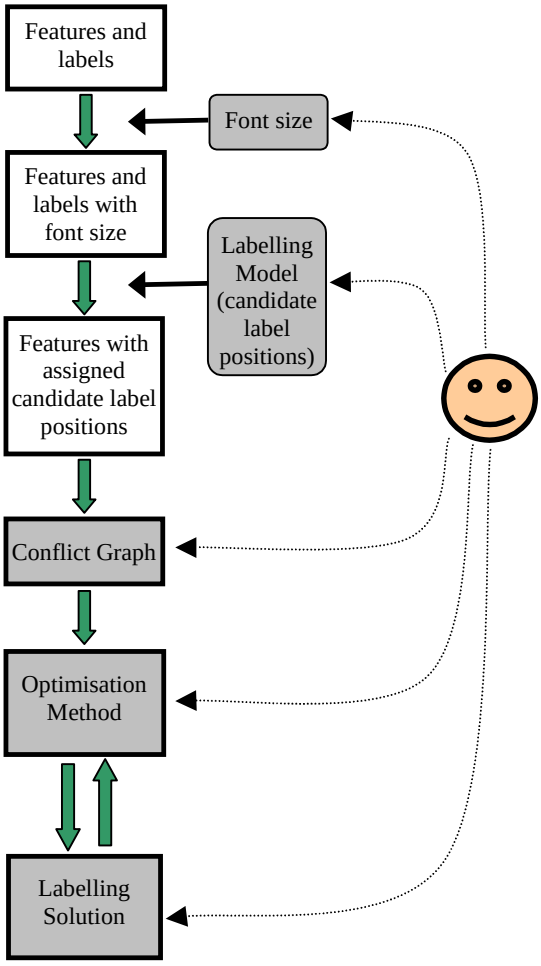
- Insertion of domain knowledge – the user should be able to continuously refine the labelling problem by inserting domain knowledge.
- Use of automatic optimisation tools – instead of performing very time-demanding manual adjustments of labelling solutions, the user should take advantage of automatic optimisation methods. The methods must compute a solution for the problem possibly considering recent domain knowledge entered into the system.
- Quality feedback – visualisation tools should provide feedback to the user about the quality of the current labelling solution and the progress of the optimisation process.

These facilities are implemented in our framework as a cyclic process where an initial labelling solution can be continuously improved. The user inserts domain knowledge, calls optimisation methods for improving the solution, and visualises the result. The process is repeated until the user is happy with the quality of the labelling.

Next, we divide the labelling process into a sequence of small steps, and show how user interaction can insert domain knowledge and control the optimisation in the several stages. The division of the labelling process in steps is based on a combinatorial approach proposed by Edmondson *et al.* (1996).

3.1 Labelling steps

Picture 3 shows a diagram of the labelling steps. The labelling process starts with a list of point features with their XY-coordinates in the map and their labels. A set of graphic attributes is then assigned to the labels, describing the font size used for writing the text. In the next step, a labelling model is incorporated into the process. We use the fixed position model described in Section 1. Up to nine predefined candidate label-positions can be assigned to each point feature. The candidate positions are rectangular regions in the map around the features. The size of the rectangular regions depends on the text and on the font size of the label they represent.



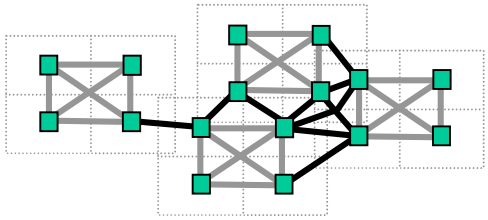
Picture 3: Labelling steps and user interactions.

After a set of candidate positions has been created, we model the labelling problem as an combinatorial optimisation problem in a traditional fashion (Edmondson *et al.* 1997, Strijk, Verweij, and Aardal 2000, and Zoraster 1990): we construct a *conflict graph*[2] *G=(V,E)* describing all overlaps between pairs of candidate label positions, and use it for computing a label assignment without overlaps and with minimal cost. The conflict graph is created in following way: for each candidate position *c* a unique vertex v_c is included in *V*; for every pair of label positions *c* and *d* that overlap on the map, an edge (v_c,v_d) is added to *E*, where v_c and v_d are vertices in *V* representing the positions *c* and *d*, respectively. If a candidate label position *c*, assigned to a point feature *f1*, overlaps another point feature *f2* with $f1 \neq f2$, then an edge (v_c,v_d) is inserted in *E* for every candidate label position *d* assigned to *f2*. Moreover, for every feature *f*, the candidate label positions assigned to *f* form a clique in *G*.

[2] For a background in Graph Theory, see Bondy (1976).

User preferences for the candidate label positions are described by cost values (real numbers between 0 and 1) assigned to the vertices of *V*. The higher the cost value, the lower the preference for the respective label position.

Many user actions basically manipulate the conflict graph, as we explain in Section 3.2. An example of a conflict graph is shown in Picture 4 for the candidate positions presented in Picture 1.



Picture 4: the conflict graph of the candidate label positions in Picture 1. Vertices, representing label positions, are drawn as squares. The original candidate label positions, shown as dotted rectangles, are presented here only for reference. Light-colour edges connect vertices related to candidate label positions of the same feature. Dark-colour edges indicate overlaps on the map between two candidate positions.

In the next step, the conflict graph is used for solving an optimisation problem. The most intuitive approach is to solve the Minimum Cost Maximum Independent Set Problem (MCMISP) for the graph of overlaps, as it is done by some popular map-labelling techniques (for example, Strijk, Verweij, and Aardal 2000). A solution for this problem is a subset *S* of the vertices of the graph with maximum size, for which there is no edge *(u,v)* in the graph that connects vertices *u* and *v* in *S*. Moreover, the sum of the costs of the vertices in *S* is minimised with respect to the size of S.

A solution for the MCMISP can be directly transformed to a labelling solution: the vertices in *S* indicate the candidate positions that must be used for labelling the features. If none of the vertices related to a particular feature *f* appears in S, then *f* remains unlabeled. Note that the edges in the graph ensure that no more than one candidate position is chosen for each feature, and that no overlap occurs.

The MCMISP provides a clear representation for the map labelling problem. However, this approach is not the best for an interactive system. The edges of the conflict graph are strong constraints in MCMISP, in fact too strong for user interaction. Inserting a new vertex *v* into the independent set *S* implies the removal from *S* of all vertices that conflict with *v*. This operation may cause drastic changes of the labelling solution, while a user would prefer to have smooth changes of the visualisation of the labelling.

As an alternative, we consider a slightly different problem (investigated by Edmondson *et al.* 1996), where overlaps between labels are tolerated, but are penalised in the objective function. The problem consists of choosing a subset *S* of the vertices of the conflict graph such that the number of overlapping labels plus the number of unlabeled features is minimised. Moreover, we add to the conflict graph an *unlabelled vertex* for every feature. This vertex forms a clique with the other vertices of the feature. If an unlabelled vertex appears in *S*, then its related feature is said to be unlabelled. All features must have exactly one of their related vertices in *S*.

The objective function includes the cost of the vertices in *S* as a secondary criterion.

With this new problem, the user is allowed to change the labelling solution in a way that causes overlaps. Nevertheless, a worse solution is expected to be an intermediate state to a better labelling.

3.2 Interactions

The user can interact with the labelling process in several ways. These interactions are shown in Picture 3 and described below.

a) Changing the font size of the label of a feature.

b) Activating/deactivating a predefined candidate position *c* for a feature – this action can be implemented as inserting/removing a vertex v_c into/from the conflict graph, plus the edges indicating overlaps with v_c.

c) Creating a customized candidate position for a particular feature. The new candidate position can be located at anywhere in the map – this action causes the inclusion of a new vertex and of some edges in the conflict graph.

d) Changing the cost of a candidate label position – the changes are made on the conflict graph.

e) Directly creating a new edge in the conflict graph or removing an existing one.

f) Forcing a feature to be labelled, independently of whether it causes other features to be unlabelled – this action is implemented by temporarily deleting the "unlabeled" vertex assigned to the feature in the conflict graph.

g) Constraining a feature *f* not only to be labelled, but also to use a particular candidate label position *c*. In this case, all vertices in the conflict graph related to *f* are temporarily removed, except the one representing *c*; the affected edges are also removed.

h) Choosing an optimisation method (when there is more than one method available for solving the labelling problem).

i) Focusing the optimisation method on a particular group of features whose labelling needs major improvement. Focus is implemented by allowing the user to select a group of features. When the optimisation method is executed, only the selected features have their labelling solution modified.

j) Manually changing the labelling solution, that is, changing the set *S* of candidate label positions computed by the optimisation method.

Actions performed by the user in a step of the labelling process are propagated to the later steps. For instance, increasing the font size of the label of a feature causes all candidates label positions assigned to this feature to be enlarged. Therefore, new overlaps may occur and new edges will be added to the conflict graph. Similarly, if the user deactivates a candidate label position c that is in use by a labelling solution S, then the related vertex v_c is removed from the conflict graph and also from S.

These interaction facilities can be classified as types of hints, according to the user-hint framework described in Section 2. The interactions (a) to (g) are adjustments of the objective function and the constraints of the problem, since they directly affect the conflict graph. Facilities (i) and (g) correspond to controlling the optimisation method. Interaction (j) is a typical manual adjustment of the solution.

With such facilities the user has a considerable amount of flexibility for refining the problem and improving a map labelling solution.

3.3 Optimisation methods

Practically any optimisation method for map labelling can be used in the approach. This includes Simulated Annealing, Hill-Climbing algorithms, Integer Linear Programming and Genetic Algorithms. There is a great benefit in using an iterative improvement method since it can take advantage of an existing solution in order to produce a new one. In the worst case, the algorithm can use the existing solution as an upper bound for the quality of a new labelling.

3.4 Feature selection

We can implement feature selection (focus) in an optimisation method in two different ways:

1) Hard-coding support for selection in the optimisation algorithm; in this case, the algorithm changes only vertices in S that are assigned to selected features.

2) Creating a small conflict graph containing only vertices and edges related to selected features. The optimisation algorithm is applied on the small graph. The resultant solution is mapped back to a solution of the whole labelling.

We have experimented both options. There are some optimisation algorithms for which it is easy to hard-code selection. The second option, however, is more interesting since it allows using a wider variety of existing map labelling algorithms without modification.

3.5 Visual feedback

Pictures of the labelling solution are provided to the user to emphasize some of its qualitative aspects and/or the combinatorial characteristics of the problem.

The user can swap between two types of pictures that are useful for this problem:

1) A geographic map with features and their labels; the picture can optionally show the set of candidate positions for every feature (as in Picture 1).

2) The conflict graph with all vertices in S highlighted.

The pictures highlight overlaps and unlabelled features by using different colours and shapes.

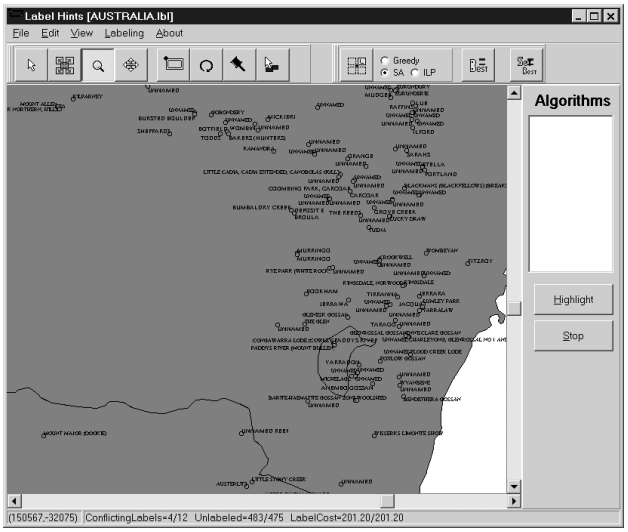
In order to avoid overwhelming the visualization with too many details, we adopt an *information-demand policy*: the user can control the amount of extra information to be displayed. This is done by combining the visualization tool with the feature-selection mechanism, so that extra details are shown only for the selected features.

4 Remarks

4.1 System Implementation

We implemented a system for experimenting with the map labelling approach. See picture 5.

At the beginning of the labelling process, the system creates a trivial solution by producing a set S with an unlabelled vertex for every feature. This is shown in the visualization by drawing crosses representing unlabelled features. The user then selects a group of features and adds candidate positions for them. Picture 6 shows a dialog window for setting the features' attributes. Next, the user calls an optimisation method for computing a better labelling solution.

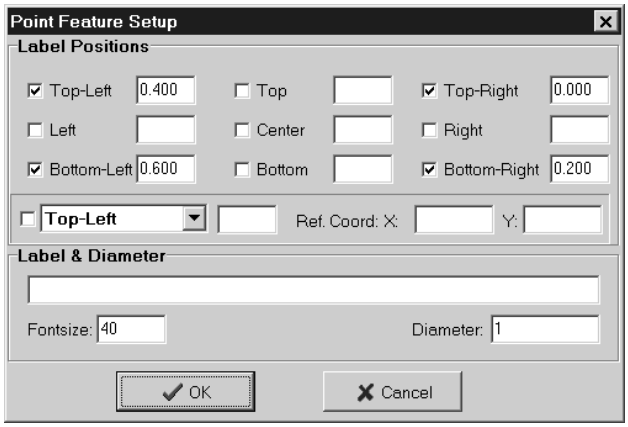


Picture 5: interactive map labelling system.

We have coded two optimisation methods – a hill-climbing algorithm and a simulated annealing algorithm; each produces good quality labelling.

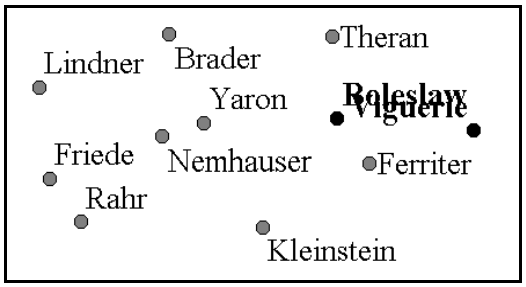
The algorithms were implemented as independent threads that can be run at any time and are controlled by the user. Several instances of the algorithms can be executed simultaneously, working on the same region or on different areas of the labelling. A small conflict graph is created for each thread and the partial solutions are integrated into the complete labelling solution. However,

there is no automatic lock mechanism of the global solution. This means that the user is responsible for deciding what algorithms to run and what labelling region to focus on. On the other hand, the system automatically computes the quality of every new solution updated by the threads and retains the best global solution.



Picture 6: configuration window.

The quality of the labelling solution is shown as textual information in a status bar (see Picture 5) or by highlighting label overlaps and unlabelled features in the visualization. Picture 7 illustrates a region of a map labelling where two labels ("Boleslaw" and "Viguerie") overlap. Other details of the labelling can be displayed, such as the candidate positions for each feature, in order to help the user to understand the combinatorial nature of the problem.



Picture 7: label overlap highlighted in the visualisation.

4.2 Labelling improvement using the interactive approach

Next, we explain how the interactive facilities presented in Section 3 can be used for solving overlaps and ambiguity, and for converging to better local minima.

4.2.1 Overlaps

In general, the optimisation methods provide an initial labelling solution without overlaps, but with some unlabelled features. Overlaps are then created by the user when moving a label to a different candidate position, changing the size of the font of the label, or changing the label text. Such changes may reflect the user's subjective assessment of the current label placement.

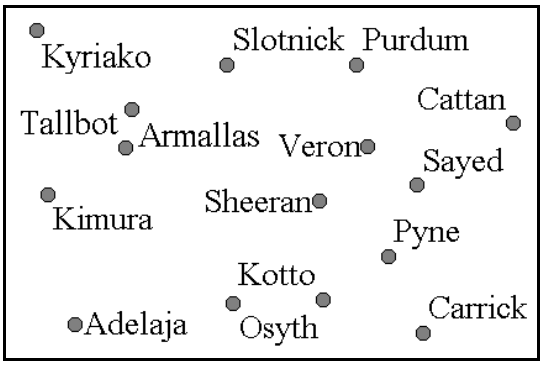
Overlaps can be solved by selecting a group of the features, including the conflicting ones, and re-executing one of the optimisation methods for reorganising the labels. In case there is no complete labelling without overlaps, then one or more features may be put in an unlabelled state.

4.2.2 Ambiguity

Ambiguity involves uncertainty in uniquely distinguishing to which feature a label is related. Picture 8 shows a crowded labelling region with ambiguous situations. For instance, it is difficult to identify the features for "Kotto" and "Osyth". The observer may also be unsure, at a first glance, about the features with labels "Slotnick" and "Tallbot".

The interactive framework offers the user three ways of reducing ambiguity:

1) deactivating one of the candidate label positions in use that is causing ambiguity;
2) increasing the cost for a pair of ambiguous candidate positions; or
3) inserting an edge in the conflict graph connecting a pair of vertices in ambiguous candidate positions.



Picture 8: ambiguous labelling.

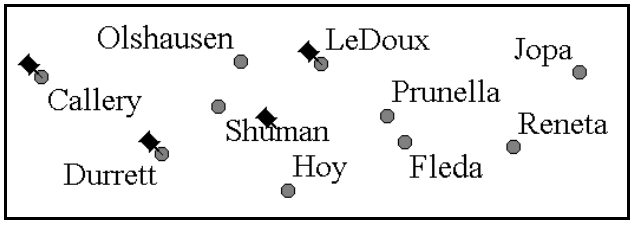
All these operations can be easily performed in the graphical interface, by a few simple clicks. Then the user calls an optimisation algorithm for computing a new solution.

Among the three alternatives for solving ambiguity, the third one seems to be most attractive. It creates a "non-visual" overlap condition between candidate positions, that can be solved as a normal overlap (Section 4.2.1).

Note that the same approach can be applied to the ambiguous case in Picture 2: the user can create a conflict edge between labels "St. George" and "Washington", and between "Washington" and "Happy Bay".

4.2.3 Unlabelled features

If an optimisation method tends to leave a particular feature unlabelled, then the user may force it to be always labelled. This is visually performed by attaching a pin to the feature (or a label position). See Picture 9. Before running the method, such a constraint is automatically transferred to the conflict graph, as explained in Section 3.2.



Picture 9: Pinned (constrained) features.

4.3 Other resources

In practical labelling tasks, the user commonly adjusts the labels by moving them on the map. We also allow the same type of interaction, but we integrate it with the combinatorial framework. Basically, a manual adjustment of a label causes a customised label position with zero cost to be created and inserted into the conflict graph. The solution *S* is then updated to have this position as the choice for the placement of the label. In future executions of a labelling algorithm, the customised position is also taken into consideration.

Another point to be noted is that the user can increase or decrease the font size of a single label or of several labels simultaneously. This allows an interactive search for the greatest label size with maximum the number of labelled features. The Maximum Label-Size Optimisation Problem is NP-Hard (see Neyer 2001 for a general description of this problem).

4.4 Limitations

The conflict graph has its limitations, which are related to the fact that it describes a simple binary relation between label positions. Therefore the conflict graph cannot be used to exclude a particular placement involving three or more label positions without also excluding partial configurations of these positions.

5 Conclusion

In this paper, we showed a list of interaction facilities for Map Labelling. The labelling approach is embedded into an interactive framework based on user hints. The user can refine the labelling problem by inserting domain knowledge, control the labelling algorithms and perform manual adjustments of an existing solution. A conflict graph of overlaps is used for representing the user's domain knowledge.

We presented the approach for the problem of labelling point-features with rectangular labels. However, the combinatorial model applied here can be extended to handle any type of feature and label shapes.

We are currently enlisting experts in Map Labelling for an extensive evaluation of the approach. The aim is to investigate how helpful each type of interaction is for solving many map labelling problems.

6 References

ANDERSON, D., ANDERSON, E., LESH, N.B., MARKS, J.W, MIRTICH, B., RATAJCZACK, D. and RYALL, K. (2000): Human-Guided Simple Search. *National Conference on Artificial Intelligence (AAAI)*, ISBN 0-262-5112-6.

BARBOSA, H. J. C. and BARRETO, A. M. S. (2001): An interactive genetic algorithm with co-evolution of weights for multiobjective problems. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 203-210, Morgan Kaufmann Publishers, San Francisco, California.

BEASLEY, J. E. (1985): Algorithms for unconstrained two-dimensional guillotine cutting. Journal of Operational Research Society Ltd, 36(4): 297-306.

BONDY, J. A. and MURTY, U. S. R. (1976): Graph Theory with applications. Macmillan: London.

CHAZELLE, B. and 36 co-authors (1999): The computational geometry impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, Advances in Discrete and Computational Geometry, vol. 223, pp. 407-463. American Mathematical Society, Providence.

CHRISTENSEN, J., MARKS, J. and SHIEBER, S. (1994): Placing text labels on maps and diagrams. In Paul Heckbert, editor, Graphics Gems IV, pp. 497-504. Academic Press, Boston, MA.

CHRISTENSEN, J., MARKS, J. and SHIEBER, S. (1995): An empirical study of algorithms for point-feature label placement. ACM Transactions on Graphics, 14(3):203-232.

EDMONDSON, S., CHRISTENSEN, J., MARKS, J. and SHIEBER, S.(1997): A general cartographic labeling algorithm. Cartographica, 33(4):13-23.

GRADIŠAR, M., JESENKO, J. and RESINOVIČ, G. (1997): Optimization of roll cutting in clothing industry. *Computers Operations Research, 24(10): 945-953*. Elsevier Science Ltd.

IMHOF, E. (1962): Die Anordnung der Namen in der Karte. In International Yearbook of Cartography, pp. 93-129, Bonn Bad Godesberg. Kirschbaum.

IMHOF, E. (1975): Positioning names on maps. The American Cartographer, 2(2): 128-144.

JACOBSEN, A. E. (2001). Interaktion und Lernverfahren beim Zeichnen von Graphen mit Hilfe evolutionarer Algorithmen ("Interaction and learning methods for graph layouts with the help of evolutionary algorithms"). Master Thesis, Institute AIFB, University of Karlsruhe, Germany.

LESH, N., MARKS, J. and PATRIGNAME, M. (2000): Interactive partitioning. *International Symposium Graph Drawing*, LNCS 1984, pp. 31-36.

MARKS, J. and SHIEBER, S. (1991): The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard CS.

MENDONÇA, C. F. X. (1994): A layout system for information system diagrams. PhD Thesis, Department of Computer Science, University of Queensland.

NASCIMENTO, H. A. D. and EADES, P. (2001a): Interactive graph clustering based on user hints. In the *Proceedings of the Second International Workshop on Soft Computing Applied to Software Engineering*, Enschede - Netherlands, February 8-9.

NASCIMENTO, H. A. D. and EADES, P. (2001b): User hints for directed graph drawing. In the *Proceeding of the 9th Graph Drawing Conference (GD 2001)*, LNCS, vol. 2265, pp. 205-219.

NASCIMENTO, H. A. D. (2001c): A framework for human-computer interaction in directed graph drawing. In the *Proceeding of the Australian Symposium on Information Visualisation*, Sydney, December, pp. 63-69.

NEYER, G. (2001): Map labeling with application to graph drawing. LNCS 2025, pp. 247-273, Spring-Verlag, Berlin Heidelgerg.

ROSETE-SUAREZ, A., SEBAG, M. and OCHOA-RODRIGUEZ, A. (1999). A study of evolutionary graph drawing. Technical Report. URL: *citeseer.nj.nec.com/411217.html*.

RUSSELL, G. (2002): Private communication. UBD, Universal Press, Australia.

SCOTT, S., LESH, N. and KLAU, G. (2002): Investigating Human-Computer Optimization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, ISBN: 158113-453-3, pp. 155-162.

STRIJK, T., VERWEIJ, B. and AARDAL, K. (2000): Algorithms for maximum independent set applied to map labelling. Technical Report UU-CS-2000-22, Department of Computer Science, Utrecht University.

TAKAGI, H. (2001): Interactive evolutionary computation: fusion of the capabilities of EC optimisation and human evaluation. *Proceedings of the IEEE*, 89(9): 1275-1296.

TOLLIS, I. (2002): Private communication. Sydways Publishing Pty. Ltd., Australia.

WOLFF, A. (1996). The map-labelling bibliography. URL:http://www.math-inf.uni-greifswald.de/map-labeling/bibliography/.

YOELI, P. (1972): The logic of automated map lettering. *The Cartographic Journal*, 9:99-108.

ZORASTER, S. (1986): Integer programming applied to the map label placement problem. *Cartographica*, 23(3):16-27.

ZORASTER, S. (1990): The solution of large 0-1 integer programming problems encountered in automated cartography. *Operating Research*, 38(5):752-759.