From Point Feature Label Placement to Map Labelling

Maxence Laurent* Éric D. Taillard† Oliver Ertz* Francis Grin‡ Daniel Rappo*
Sébastien Roh‡

*Institute of Information and Communication Technologies, HEIG-Vd
Route de Cheseaux 1, CP, CH-1401 Yverdon-les-Bains, Switzerland
maxence.laurent, olivier.ertz, daniel.rappo@heig-vd.ch

†Institute of eMbedded Information Systems, HEIG-Vd
Route de Cheseaux 1, CP, CH-1401 Yverdon-les-Bains, Switzerland
eric.taillard@heig-vd.ch

‡Geomatics, Environment management, Building and work supervising Instiute, HEIG-Vd
Route de Cheseaux 1, CP, CH-1401 Yverdon-les-Bains, Switzerland
francis.grin, sebastien.roh2@heig-vd.ch

1 Introduction

With the fast development of geographic information systems (GIS), map labelling is increasingly important. A map is drawn with 3 different types of elements:

- Points, representing top of montains, isolated buildings, small villages, etc
- Lines, representing rivers, streets, etc.
- Polygons, representing lakes, districts, large rivers, building at small scale, etc.

Some of the objects should be labelled by a text that should be placed close to it. In the case of labelling a point-feature, the text is placed around the object (and not directly above it, to avoid obscuration of the feature). The goal of point-feature labelling is to find a position for each label in such a way that no label overlaps another one or overlaps the symbol marking a point. Very often, finding a conflict-free labelling is not possible. So, a labelling containing as many label as possible without conflicts is searched for. As soon as more than one candidate position is possible for placing the label of each object, the problem of choosing an adequate position for each label is difficult (NP-Hard). Basically, the problem is modelled as finding a stable set of maximum cardinality in a conflict graph. The conflict graph is built as follows : a vertex is associated to each candidate label and there is an edge connecting two vertices if the associated labels cannot be placed simultaneously on the map, either because they overlap or they label the same object.

Efficient metaheuristics have been developed recently for finding good solutions to point-feature labelling. Among the fastest ones providing the highest quality solutions, let us quote the POPMUSIC approach of [1]. The computational complexity of this method grows quasi-linearly with the number of labels and quadratically with the label density (that can be measured with the degree of the vertices of the conflict graph). Numerous other methods have been proposed for this problem; the reader is referred to [7] for an extensive bibliography.

The goal of this paper is manifold:

- Showing how to extend the work of [1] for being able to include line- and polygon-features
- Presenting a chained neighbourhood for on-line labelling
- Presenting the *PAL* open-source library for map labelling

2 The labelling process

Producing a map involves a number of steps. First of all, the data should be collected and stored in databases. The data are organized in *layers*. Each layer contains homogeneous data, for instance, there is one layer containing all roads, another one all lakes, etc. These data can be edited and exploited using software like $Mapinfo$, *ArcGIS* (among the most popular proprietary (and very expensive) codes) and *GRASS*, *QuantumGIS*, *GvSIG* (among the open-source software).

Once the appropriate layers have been accessed with a GIS software, the user has to extract the portion of the map that has to be treated ($MapExtent$). Concerning the labelling process, the user has to specify how each layer must be treated.

- The layer has just to be drawn on the map, without being labelled. In this case, the objects of the layer may either be considered as *obstacles*, i.e. zones where no label should be placed (if possible) or the objects may be covered by labels of other layers.
- The object of the layer has to be labelled. In this case, the objects may either be considered as obstacles for the labels of other layers or not.
- The priority of each layer, in order to favour the placement of the most important labels.
- The arrangement of the labels around or above the object drawing.

There are several ways to label an object. The label can be either placed at a limited number of fixed positions relatively to the object (discrete, fixed position model) or continuously around the object (slider model). In practical labelling, it is generally not allowed that a label overlaps another one, and there are preferences among the potential position of a label for a given object. The optimization methods developed in the present paper considers the discrete model, takes the preferences into account and try to maximize the number of labels displayed without overlap. It is assumed that the labels are rectangular with a dimension (width $\times$ height) that depends on the object to qualify. The dimension of a label depends neither on its actual position nor on the position of other labels. So, PAL library does neither consider labels that follows a broken line (such

as a river) nor labels that are written in several parts (such as long names, large objects written with very large fonts). However the orientation of the label may eventually be adapted. Therefore, a solution to a map labelling problem is provided as a list of 4 characteristics for each object: labelled or not, position of the left-bottom corner of the label ((x, y) coordinates) and orientation ($-90° < \alpha \leq 90°$).

3 Generation of label candidate positions

The possible positions for the label of an object depend on the object type (point, line, polygon) and on preferences specified by the user.

3.1 Label candidates for a point-feature

On a map, a point feature object is drawn by a symbol (circle, star, square, etc.) that can be included in a circle of radius r. The label of this object cannot overlap with this circle. The candidate positions for a point feature are spread as regularly as possible around this circle (see Figure 1). Point-features are almost always labeled horizontally in practice.

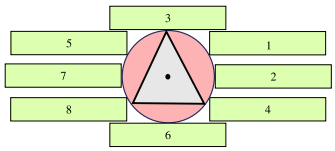

Figure 1: First order triangulation point in Swiss national maps (triangle with a dot), circle of radius r including this symbol and 8 candidate possible positions for labelling this object. Generally, the size of the label is much larger than the point-feature.

3.2 Label candidates for line-feature

Linear-feature objects (brooks, small roads) are represented with broken lines in vector databases ($shapefile$). In practice, the label associated with a line is almost always oriented in a direction locally parallel to the line. Since we only consider rectangular labels, the orientation of the label is determined as follows: Let w be the width of the label, and A and B two points of the line at a distance of w each others (the distance being measured along the broken line). The orientation of the label is parallel to the segment $\overline{AB}$. The user can specify two ways of labelling a line: either the label is placed on the line (and the segment $\overline{AB}$ is right on the centre of the label) or the label is on the one side or the other of the line (the segment $\overline{AB}$ is at distance r of the label). Figure 2 illustrate how candidate positions are generated for line-features.

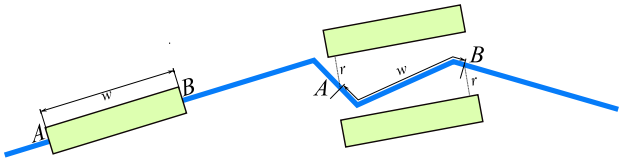

Figure 2: Possible labelling of a broken line: the label is placed on the line or on the one side or the other at distance r

3.3 Label candidates for polygon-feature

The user can specify different way for labelling a polygon. First, a (small) polygon can be considered as a point-feature. The candidate generation is done as for point-features in that case. Second, the label can be placed on the border of the polygon. The polygon is therefore considered as a broken line and the user can specify the both ways for labelling line-features. Finally the user can specify that the candidate positions must be inside the polygon (in case the last is sufficiently large; otherwise, the label is only partly inside). For positions that are inside a polygon, the orientation of the label can be specified to be either horizontal or freely chosen by the system. In the last case, the polygon is decomposed into a set of (almost) convex polygons. Each of these convex polygon is embedded into a rectangle box of minimal surface. The orientation of the label is parallel to the border of this box. Figure 3 illustrates how candidate positions are generated for a non-convex polygon.

4 Candidate positions filtering

The candidate positions generation process described above may be applied to map labelling with the slider model. Since we use algorithms tailored for the discrete model and since these algorithms are very sensitive to the degree of the nodes of the conflict graph, the number of candidate positions for each object must be limited. The computational effort of these algorithms grows quadratically with the number of conflicts. So, the user specifies for each object the maximal number of candidate positions to retain. Practically, less than a dozen of candidate positions should be generated for point-features and few dozen for line- and polygon-features.

For generating the number of candidates specified by the user, we proceed as follows: The simplest case is for point-features. The candidates are uniformly distributed around the circle of radius r centred on the object. The quality of a candidate depends on its position. It is preferred to place a label above and on the right of the object rather than below and on the left. The 8 candidate positions of Figure 1 are numbered by decreasing preference.

Concerning line-feature objects, candidate positions are generated at distance w along the broken line, where w is the length of the label. A quality coefficient is attributed for each candidate position. This coefficient depends on the length $\|\overline{AB}\|$ of the segment used to give an orientation to the candidate. The closer $\|\overline{AB}\|$ from w is, the better the quality. If $\|\overline{AB}\| = w$, the orientation is exactly parallel to the line.

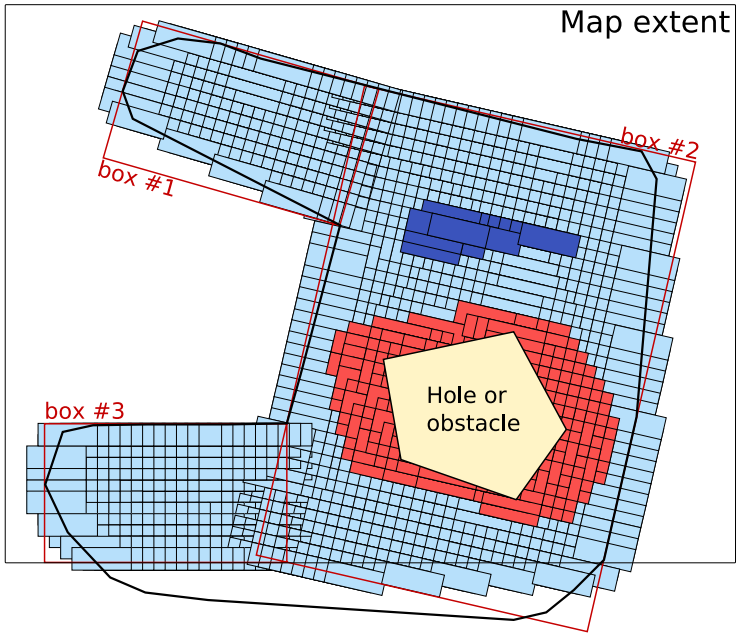

Figure 3: Possible labelling of a non-convex polygon: the polygon is decomposed into a set of convex ones which are embedded into rectangular boxes. The orientation of the label candidates is parallel to the border of these boxes. The polygon may contain a hole or other type of features that should be considered as obstacle. The candidates are filtered so that they are at maximal distance from the border of the polygon or from an obstacle (in dark).

For polygon-feature objects, the candidates are positioned on a grid whose mesh size is $\min(w/2, h/2)$ where w is the width of the label and h is the height. Only the candidates that intersect the polygon are retained. Then, a quality measure is attributed to each candidate position. If the last is partially outside the polygon or if it intersects the object of another layer that has to be treated as obstacles, the quality factor is minimal. Otherwise, the quality factor is inversely proportional to the distance with the polygon border or other obstacles. Among all the candidates generated for a polygon, only the best one are retained. In Figure 3, the candidate label in red are first removed and those in dark blue have the highest quality.

5 Optimization techniques

The optimization techniques implemented in PAL library consider that two labels cannot overlap as a hard constraint. Therefore, if an object cannot be labelled without conflict, a non-labelled penalty that depends on the object is taken into account in the objective function. For the objects that are labelled, the quality of the label position retained is taken into consideration. All the methods implemented in PAL library try to minimize this penalty function under the constraint that no label can overlap.

5.1 Initial solution

An initial solution is obtained by applying the first step of the FALP method [9]. This step is a greedy algorithm that retains the candidates in decreasing order of the number of conflicts with other candidates of objects not yet labelled. The remaining steps of this method have not been applied since they produce unfeasible solutions with overlaps and their time complexity is too high for labelling large real maps.

5.2 Elementary neighbourhood

All improving methods implemented in the PAL library are based on an elementary neighbourhood consisting of:

- Selecting a candidate label for an object not labelled
- Suppressing the label of a labelled object
- Choosing another candidate label for an already labelled object

This is an extension of the neighbourhood used in the method of [1] to take into account that the non-overlapping constraint is a hard one.

5.3 Chained neighbourhood

On the basis of the elementary neighbourhood, an extended one has been designed, using the principle of ejection chains [3]. The chain is initialized either by adding a new label L for an object not yet labelled or by modifying the position of a label of an object already labelled. Such a modification may lead to three situations:

- The new label L does not overlap with another one. In this case L is retained and the ejection chain stops.
- The new label L overlaps with 2 or more other labels. In this case L is removed and the ejection chain stops.
- The new label L overlaps with exactly one other label. In this case, another position is chosen for this other label. The position has to be chosen in such a way that it does not overlap with L or with a label that has been moved in the chain. If no such position exists, L is removed and the ejection chain stops. Otherwise, the position for the other label is chosen in such a way that it generates a minimum number of overlaps. The chain is then continued, unless the length of the chain is higher than a parameter (50 in our implementation).

For each (unfeasible) solution of the chain it is possible to build a feasible solution by removing the label just modified. The chained neighbourhood contains all the feasible solutions that can be so obtained. Since the chain is propagated only if one label is in conflict, no exponential explosion of the neighbourhood size occurs.

5.4 Tabu search

Two tabu searches have been integrated in PAL library: the first is based on the elementary neighbourhood and is similar to those implemented in [1]. The second tabu search uses the chained neighbourhood.

5.5 POPMUSIC

Since the goal of PAL library is to be able to produce a map labelling in a reduced amount of time and since tabu search is not very efficient for treating problem instances with more than few hundreds of objects [1, 8], POPMUSIC framework [4] is adequate for speeding up the search process. The basic idea behind POPMUSIC is to build sub-problems on the basis of a solution. The sub-problems are optimized independently and used to modify the solution. In the map labelling case, a sub-problem is built as follows: A label is first selected. Then the t objects whose candidate labels are the closest from the first one (in terms of edges in the conflict graph) are selected. These t objects form the sub-problem. The label positions of these t objects can be modified and optimized independently as soon as all candidate positions that overlap with the label of objects outside the sub-problem are removed.

Three POPMUSIC variants have been implemented in PAL library. They only differ by the method used to optimize sub-problems. The first one uses a simple improving method based on the chained neighbourhood. The second one uses a tabu search based on the elementary neighbourhood. The third one uses a tabu search based on the chained neighbourhood.

6 Numerical results

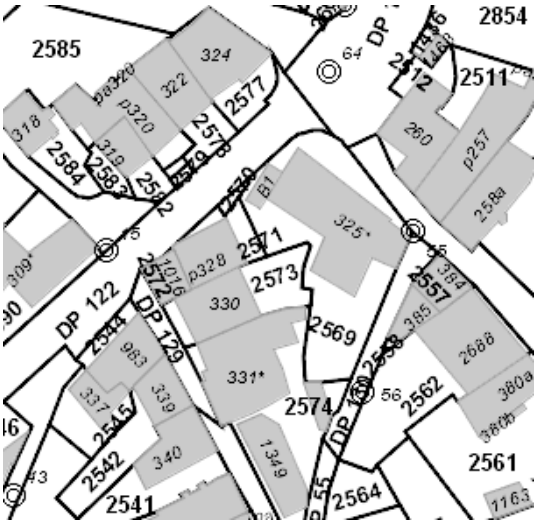

Figure 4: Map produced with *GvSIG* when a plug-in implementing the methods presented in this article is used

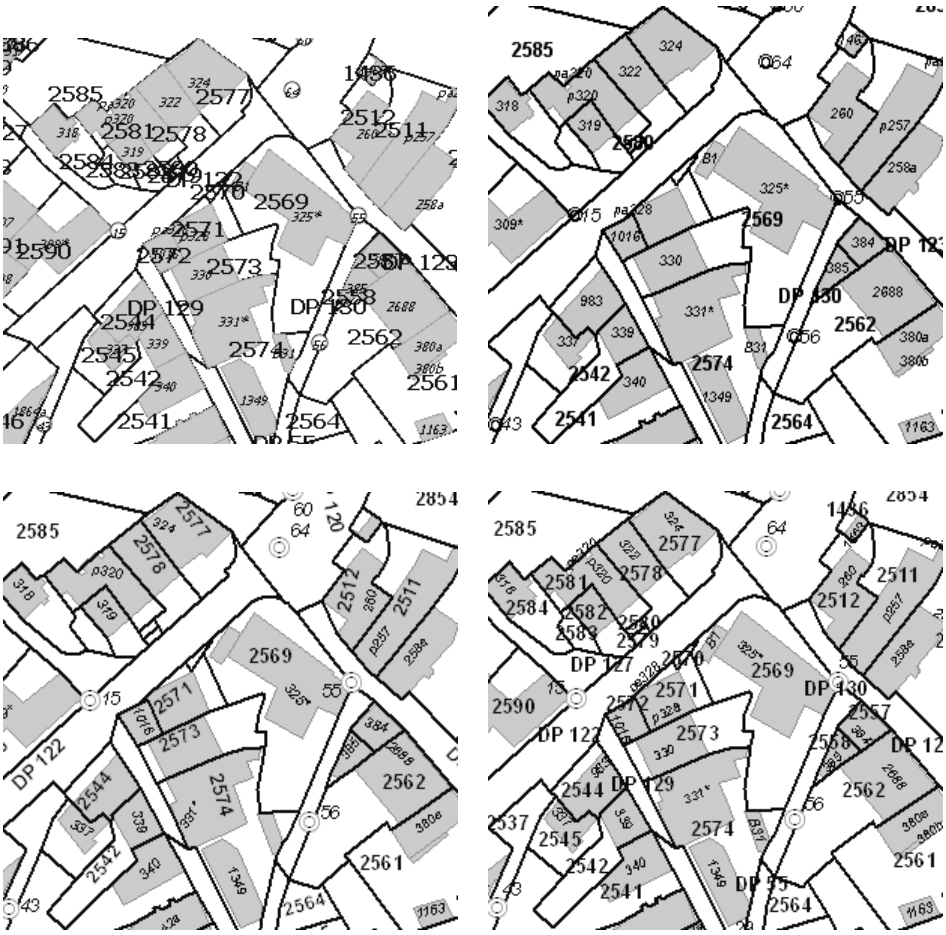

Figure 5: Map produced with the open-source software *QuantumGIS*1.0.1 (top left), with the commercial softwares *MapInfoProfessional*9.5 (top right), *ArcGIS*9.3 (bottom left) and *ArcGIS*9.3 with *Maplex*3.4 plug-in (bottom right).

Modelling the general labelling problem under the form of a fixed positions problem (production and filtering candidate label positions) as well as the tuning of the algorithms has been done while considering real problem instances. The characteristics of real problems are very different form randomly generated ones. Unfortunately, the data of the real problem instances cannot be published. So, it is not scientifically interesting to publish numerical results concerning these instances. Let us mention that instances containing 600 to 1000 objects (with a conflict graph containing 7 000 to 236 000 edges) can be tackled in a few milliseconds to few seconds, while larger instances with about 10 000 objects (half of them being polygons, one third being lines and the remaining point-features) and conflict graphs containing up to 1.5 million of edges can be tackled in few seconds to less than 2 minutes on a laptop computer. Visually, the labelling obtained with the methods presented in this article are better than those produced with standard software of the market. Figure 4 shows the labelling obtained with our method when used in the open-source geographic information system *GvSig*. The map is composed of polygons (parcels and buildings) and points (waste water accesses) for a dense region of a village in Switzerland. The labels are the reference number of the objects. A building is always placed on a parcel. For legibility, it is better to place the label of a parcel (large font) outside a building, if the last does not occupy the whole parcel.

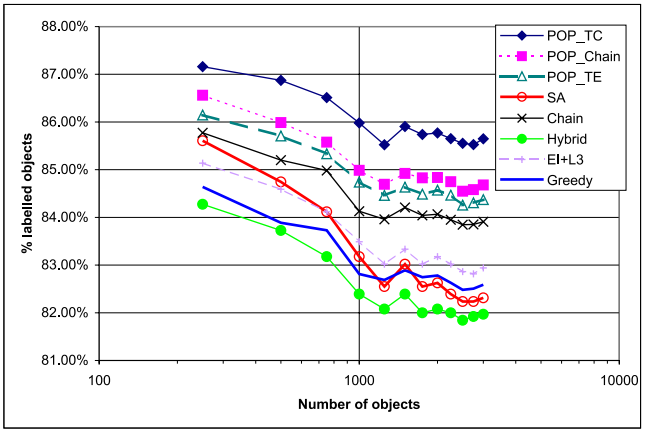

Figure 6: Solution quality produced by several method on *RandomRectangle* instances

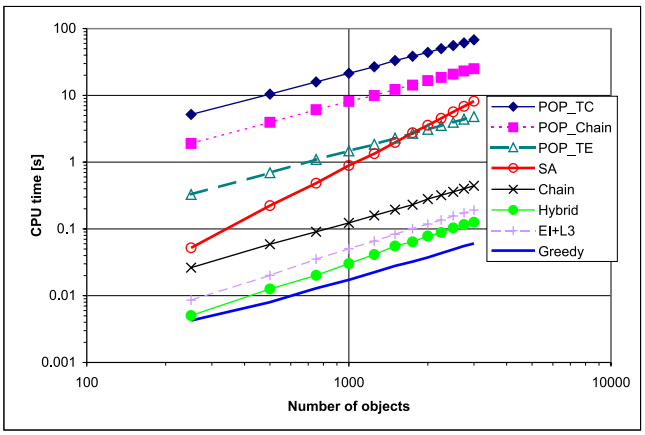

Figure 7: Computational time used by several method on *RandomRectangle* instances

We can remark in Figure 4 that improvements are still possible. However this has to be compared to automatic labellings obtained with other softwares, shown in Figure 5. We see in that figure that no label management is done by the open-source software $QuantumGIS1.0.1$. Each label is placed at the centroid of a polygon or in the middle of a point, even if it lays over another label. A large number of labels are missing in the maps produced either by $MapInfoProfessional9.5$ or $ArcGIS9.3$, quite expansive softwares ; with $MapInfo$, the labels are not oriented. When the (expansive) labelling extension $Maplex3.4$ is added to $ArcGIS9.3$, we see that almost all labels are present. However, the labels of the parcels are not oriented and are often placed over buildings.

Numerous problem instances are publically available, but most of them are provided under an incomplete form. Either the position of the objects or the exact dimensions of the label or both are unknown. Two classes of artificial problem instances proposed in [5, 6], *RandomRectangle*

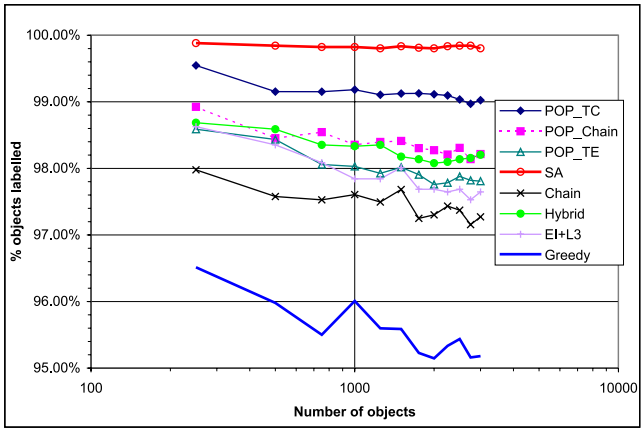

Figure 8: Solution quality produced by several method on *HardGrid* instances

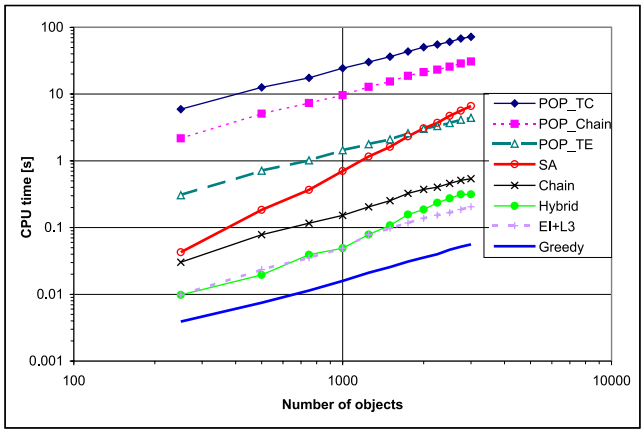

Figure 9: Computational time used by several method on *HardGrid* instances

and *HardGrid*, have been used for comparing our methods with 3 others discussed in this last reference. The number of conflicts for these instances is on the average 10 to 15 per object. We restrict ourselves to the fastest methods for which numerical results are published in [5, 6]. The methods considered are:

- Edge-Irreductibility + L3 (EI+L3) [5],
- Hybrid [5]
- Simulated Annealing (SA) [2]
- The greedy algorithm used for getting a first solution (Greedy)

- A local search improving method based on chained neighbourhood (Chain)
- POPMUSIC with a tabu search based on the elementary neighbourhood [1] (POP_TE)
- POPMUSIC with Chain local search (POP_Chain)
- POPMUSIC with a tabu search based on chained neighbourhood (POP_TC)

Several other time consuming techniques have been proposed in the litterature. Generally, they provide solutions with quality worse than those discussed in the present article [1]. Figures 6 to 9 show the average proportion of objects (points) that are labelled and the computational time as a function of the number of objects of the instance. The computational times of the first 3 methods listed above should be considered as indicative since the algorithms were run on different computers and programmed by different authors. We have estimated the relative speed factor of these computers and we have normalized the time to our computer (2.4Ghz Intel Core2 E6600).

In these figures we remark that SA produces the best results for *HardGrid* but this method has two major drawbacks: first it is not robust (it produces almost the worst results for *RandomRect* instances) and the computational effort grows quadratically with the number of objects while all the other methods proposed here behave quasi-linearly. The other methods provide generally consistent results. The higher the computational effort is, the better the quality of the solutions produced. The reader is referred to [1] for other computational results. Especially, this reference shows that POP_TE is one of the best method among several recent ones, including genetic algorithms, tabu search and Lagrangean relaxation with clusters. [8, 9] report that SA is not as efficient as other recent methods.

7 Conclusions

This article presents how to use algorithms developed in the context of a fixed positions model for point-feature labelling to real map labelling. However, an important effort has been undertaken to model the problem in order to produce instances that are able to be solved by the algorithms. Indeed, the lasts are very sensitive to the degree of the conflict graph. Another contribution of this article is to introduce the chained neighbourhood and its use in a POPMUSIC framework. This allows proposing robust and fast methods.

The C++ source codes of PAL library are publically available on `http://geosysin.iict.ch/pal-trac/`. A compiled version is also available as plug-in for the free GvSIG software `http://www.gvsig.org/web/plugins/downloads/pal-automated-placement-of-labels/` or `http://geosysin.iict.ch/trac/wiki/Index4extJPAL/`. The code is going to be integrated in QuantumGIS `http://www.qgis.org/`.

References

[1] Adriana C. F. Alvim and Éric D. Taillard. POPMUSIC for the Point Feature Label Placement. *European Journal of Operational Research*, 192:396–413, 2009.

[2] Christensen J., Marks J., and Shieber S. An Empirical Study of Algorithms for Point-Feature Label Placement. In: *ACM Transactions on Graphics* **14**, 203–232, 1995.

[3] Glover Fred, and Laguna Miguel *Tabu Search*. Kluwer Academic Publishers, Boston, USA, 1990.

[4] Taillard Éric, and Voss Stefan. POPMUSIC: Partial Optimization Metaheuristic Under Special Intensification Conditions". In: Ribeiro, C. and Hansen, P. (eds.): *Essays and surveys in metaheuristics*. Kluwer Academic Publishers, Boston, USA, 613–629, 2001.

[5] Wagner F., Wolff A., Kapoor V. and Strijk T. "Three rules suffice for good label placement". *Algorithmica* 30, 334–349, 2001.

[6] Wolff A. "Automated label placement in theory and practice". Ph. D. Dissertation, *Fachbereich Mathematik und Informatik*, University of Berlin, 1999.

[7] Wolff A. The Map-Labeling Bibliography.
`http://i11www.iti.uni-karlsruhe.de/~awolff/map-labeling/bibliography/`.

[8] Yamamoto M., Camara G. and Lorena L.A.N. Tabu Search Heuristic for Point-Feature Cartographic Label Placement. In: *GeoInformatica* **6**, 77–90, 2002.

[9] Yamamoto M., Camara G. and Lorena L.A.N. Fast Point-Feature Label Placement Algorithm for Real Time Screen Maps. Presented at *VII Brazilian Symposium on GeoInformatics* (GEOINFO 2005), Campos do Jordo, Brazil, November, 2005.