# POPMUSIC pour le placement de légende sur des plans

Éric TAILLARD[1], Gregory BURRI[1]

*1. EIVD, Route de Cheseaux 1, CP, CH-1400 Yverdon-les-Bains, Suisse*

eric.taillard [at] eivd.ch

**Mots-clefs :** placement de légendes, métaheuristique, méthode de décomposition

Le problème du placement automatique d'étiquettes sur une carte de géographie est de plus en plus important dans le contexte de l'affichage à l'écran de portions de cartes de navigation. Le problème consiste à choisir la position de chaque légende des objets à qualifier de sorte que la carte reste lisible tout en fournissant le plus d'information possible. En particulier, on cherche à minimiser le nombre de légendes se chevauchant tout en maximisant leur placement dans une position la plus favorable à la lecture (par exemple, au dessus et à droite de l'objet à étiqueter).

Mathématiquement, le problème du placement de légendes est NP-difficile et peut se formuler comme la recherche d'un ensemble stable de poids optimal dans un graphe. Ce dernier est construit ainsi : pour chaque objet $i$ à qualifier, on considère $p_i$ positions admissibles, associées à des sommets $v_{i_1}, v_{i_2}, \ldots, v_{i_{p_i}}$. Soit $V_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{p_i}}\}$. L'ensemble $V$ des sommets du graphe est constitué par $V = \cup_{i=1}^{n} V_i$, où $n$ est le nombre d'objets à étiqueter. Deux sommets $u, v \in V$ seront reliés par une arête si les légendes $u$ et $v$ ne peuvent figurer simultanément sur la carte, par exemple parce que $u$ et $v$ font partie du même ensemble $V_i$ (deux légendes pour le même objet $i$) ou parce que les légendes $u$ et $v$ se chevauchent. On associe à chaque sommet $v_{ij}$ un poids $w_{ij}$ correspondant à la qualité du placement de la légende de l'objet $i, (i = 1, \ldots, n)$ en position $j, (j = 1, \ldots, p_i)$.

Parmi les meilleures méthodes actuellement disponibles pour le problème du placement de légendes, on peut citer la recherche avec tabous de [5] ainsi que l'algorithme génétique de [6]. Le présent article intègre la recherche avec tabous de [5] dans une trame de POPMUSIC [3]. Le pseudo-code de cette trame peut s'exprimer de la façon suivante :

1. Données :
   (a) Une solution $s$ composée de $p$ *parties* $s_1, \ldots, s_p$
   (b) Un paramètre $1 < r \leq p$ entier
   (c) Une procédure d'optimisation
2. Algorithme
   (a) $O$ : ensemble de parties de la solution, initialement vide.
   (b) Tant que $O$ ne contient pas toutes les parties, répéter :
      i. Choisir une partie $s_i$, $s_i \notin O$.
      ii. Créer un sous-problème $R$ comportant les $r$ parties *les plus proches* de $s_i$.
      iii. *Optimiser* le sous-problème $R$.
      iv. Si la solution du sous-problème $R$ a pu être améliorée alors :
         A. Mettre à jour la solution globale.
         B. $O = \emptyset$
      v. Sinon (la solution n'a pas été améliorée)
         A. $O = O \cup s_i$

Pour implanter un algorithme basé sur cette trame, il convient de préciser ce que sont les concepts écrits en italiques : partie de solution, distance entre deux parties et procédure d'optimisation.

Pour le placement de légendes, une partie d'une solution peut être définie par un objet à qualifer, ou, dans son expression sous forme de graphe, par l'ensemble $V_i$ des sommets-légendes associés à

| **Algorithme** | **100** | **250** | **500** | **750** | **1000** |
|---|---|---|---|---|---|
| POPMUSIC | 100.0 | 100.0 | 99.6 | 97.4 | 92.3 |
| POPMUSIC version 2 | 100.0 | 100.0 | 99.5 | 97.2 | 91.6 |
| $\text{CGA}_{best}$ | 100.00 | 100.00 | 99.6 | 97.1 | 90.7 |
| $\text{CGA}_{average}$ | 100.00 | 100.00 | 99.6 | 96.8 | 90.4 |
| Tabu search | 100.00 | 100.00 | 99.2 | 96.8 | 90.00 |
| GA with masking | 100.00 | 99.98 | 98.79 | 95.99 | 88.96 |
| GA | 100.00 | 98.40 | 92.59 | 82.38 | 65.70 |
| Simulated Annealing | 100.00 | 99.90 | 98.30 | 92.30 | 82.09 |
| Zoraster | 100.00 | 99.79 | 96.21 | 79.78 | 53.06 |
| Hirsh | 100.00 | 99.58 | 95.70 | 82.04 | 60.24 |
| 3-Opt Gradient Descent | 100.00 | 99.76 | 97.34 | 89.44 | 77.83 |
| 2-Opt Gradient Descent | 100.00 | 99.36 | 95.62 | 85.60 | 73.37 |
| Gradient Descent | 98.64 | 95.47 | 86.46 | 72.40 | 58.29 |
| Greedy | 95.12 | 88.82 | 75.15 | 58.57 | 43.41 |

TAB. 1 – Pourcentage d'étiquettes placées sans chevauchement pour les exemples de problèmes de `http ://www.lac.inpe.br/~lorena/instancias.html`

l'objet $i$. La distance entre deux parties $V_i$ et $V_k$ pourra être définie comme le nombre minimum d'arêtes d'une chaîne entre des sommets de $V_i$ et $V_k$. La procédure d'optimisation utilisée est une réimplantation de la recherche avec tabous [5], qui a été calibrée pour résoudre au mieux des sous-probpèmes avec environ 70 légendes.

Nous avons considéré deux variantes de POPMUSIC : la première est telle que décrite ci-dessus et la seconde, plus rapide, n'inclut pas uniquement $s_i$ dans $O$ lorsqu'on n'a pas réussi à améliorer un sous-problème $R$ mais toutes les parties de $R$ (i.e. $O = O \cup R$).

Le tableau 1 compare la qualité des solutions obtenues à l'aide des deux versions de POPMUSIC sur des jeux de problèmes-tests comprenant entre 100 et 1000 objets. Pour être compatible avec les résultats compilés dans [6] nous donnons le pourcentage d'étiquettes placées sans chevauchement, et non la valeur des solutions en termes de la somme des qualités $w_{ij}$ de placement des légendes.

Pour la version initiale de POPMUSIC, le nombre d'étiquettes constituant un sous-problème était de 40, alors que pour la version 2, les sous-problèmes comportaient 70 étiquettes. Dans les deux cas, le nombre d'itérations de la recherche avec tabous a été limitée à 70.

Dans ce tableau $\text{CGA}_{best}$ est le meilleur résultat obtenu sur 6 exécutions de l'algorithme génétique de [6], $\text{CGA}_{average}$ est le résultat moyen de cet algorithme, Tabu search est la recherche avec tabous de [5], GA et GA with masking proviennent de [4], Simulated annealing, 3-opt gradient, 2-opt gradient, gradient descent et greedy proviennent de [1], Hirsch de [2] et Zoraster de [7]. On voit dans ce tableau que POPMUSIC donne des résultats de meilleure qualité que les autres méthodes en compétition, du moins pour les exemples de problèmes les plus grands. Pour obtenir ces résultats, les deux versions de POPMUSIC ont été exécutées 25 fois et le tableau donne la valeur moyenne des solutions obtenues.

Dans le tableau 2 les temps de calcul des méthodes donnant les meilleurs résultats sont comparés. En ce qui concerne POPMUSIC, le processeur utilisé était un Pentium III cadencé à 750 Mhz. Les temps indiqués sont ceux que la méthode a pris en moyenne jusqu'à la fin de son exécution. Pour ce qui est de CGA, les temps indiqués sont ceux jusqu'à l'obtention de la meilleure solution ; dans ce cas le processeur était également un Pentium III, mais avec une cadence non indiquée par les auteurs (au moins 400 Mhz ?).

La lecture de ce tableau montre que même la version la plus lente de POPMUSIC est de 1 à 2 ordre de grandeur plus rapide que les autres méthodes en compétition.

| **Algorithme** | **100** | **250** | **500** | **750** | **1000** |
|---|---|---|---|---|---|
| POPMUSIC | 0.0 | 0.0 | 0.3 | 3.5 | 20.0 |
| POPMUSIC version 2 | 0.0 | 0.0 | 0.2 | 1.3 | 4.4 |
| $\text{CGA}_{best}$ | 0 | 0.6 | 21.5 | 228.9 | 1227.2 |
| $\text{CGA}_{average}$ | 0 | 0.6 | 21.5 | 195.9 | 981.8 |
| Tabu search | 0 | 0 | 1.3 | 76.0 | 352.9 |

TAB. 2 – Temps de calcul (secondes) de quelques-unes des meilleures méthodes

## RÉFÉRENCES

[1] J. Chistenesen, J. Marks, S. Shieber, «An empirical study of algorithms for point-feature label placement», *ACM Transactions on Graphics* 14, 3, 1995, pp.203–232

[2] S.A. Hirsch, «An algorithm for automatic name placement around point data», *American Cartographer* 9, 1, 1982, pp.5–17

[3] É. D. Taillard, S. Voß, «POPMUSIC : Partial Optimization Metaheuristic Under Special Intensification Conditions», dans : C. C. Ribeiro, P. Hansen, *Essays and surveys in metaheuristics*, Kluwer academic publishers, 2002, pp.613–629

[4] O. Verner, R. L. Wainwright, D. A. Schoenefeld, «Placing text labels on maps and diagrams using genetic algorithms with masking», *Informs journal on Computing*, 9, 1997, pp.266–275

[5] M. Yamamoto, G. Camara, L. A. N. Lorena, «Tabu Search Heuristic For Point-Feature Cartographic Label Placement», *GeoInformatica* 6, 1, 2002, pp.77–90

[6] M. Yamamoto, L. A. N. Lorena, «A Constructive Genetic Approach to Point-Feature Cartographic Label Placement», *Actes de* $5^{th}$ *Metaheuristics International Conference*, Kyoto, août 2003

[7] S. Zoraster, «The solution of large 0-1 integer programming problems encountered in automated cartography», *Operations Research* 38, 6, 1990, pp.752–759