POPMUSIC for the Point Feature Label Placement

Adriana C. F. Alvim[†] Eric D. Taillard[*]

[†]Instituto Tecnológico de Aeronáutica, Divisão de Ciência da Computação - ITA/IEC
São José dos Campos, 12.228-900, SP, Brazil
adriana@ita.br

[*]Department of Electrical & Computer Engineering, University of Applied Sciences of Western Switzerland
Route de Cheseaux 1, Case postale, CH-1401 Yverdon, Switzerland
eric.taillard@eivd.ch

1 Introduction

Label placement is a problem of fundamental importance in cartography, where text labels must be placed on maps while avoiding overlaps with cartographic symbols and other labels. It requires positioning labels of area (such as countries and oceans), line (such as rivers and roads) and point (such as cities and mountain peaks) features [1]. Independent of the features being labeled these problems are NP-hard.

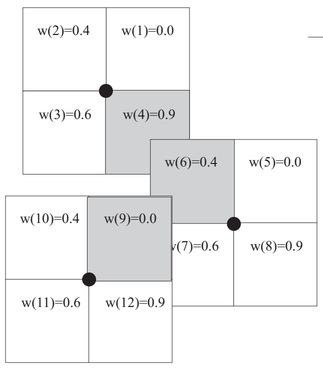
POPMUSIC [4] is a general optimization method especially designed for optimizing the solutions of large instances of combinatorial problems. The basic idea of POPMUSIC is to locally optimize sub-parts of a solution, once a solution to the problem is available. These local optimizations are repeated until a local optimum is found.

This work is based on the work of Burri and Taillard [2, 3] which investigates the evaluation of the POPMUSIC methodology to the point-feature label placement problem (PFLP) which is the problem of placing text labels adjacent to point features on a map or diagram so as to maximize legibility. The PFLP consider candidate label positions for each point feature and each label has a list of labels with which it overlaps. The objective is to place one candidate label for each point so as to minimize the number of point features which label has one or more overlaps.

In the next section we introduce the PFLP problem, Subsection 2.1 presents a tabu search approach for PFLP and Subsection 2.2 presents the POPMUSIC based heuristic for PFLP. Preliminary computational results and some concluding remarks are presented in Section 3.

2 POPMUSIC based Heuristic to PFLP

Given n points with p candidate positions for each one we have $v = n{*}p$ potential label positions represented by the integers $1, \ldots, v$. We represent each point x, $x = (1, 2, \ldots, n)$ by a variable y_x where $y_x \in \{(x-1){*}p+1, (x-1){*}p+2, (x-1){*}p+3, \ldots, x{*}p\}$. Associated with each label y_x there is a weight $w(y_x) \in \{0.0, 0.4, 0.6, 0.9\}$ which corresponds the quality of its placement, lower values indicating best positions. We are also given an overlap symmetrical $v \times v$ matrix A where $a_{ij} = 1.0 + w(j)$ if label i overlaps label j, $a_{ij} = w(i)$ for $i = j$ and $a_{ij} = 0$ otherwise. A solution S is a list of n labels $(y_1, y_2, \ldots, y_n)$. For a given solution S, a typical quality measure counts the number of point features labeled with one or more overlaps (same as the number of labels with conflicts) and is expressed by $f(S) = \sum_{i=1}^{n} \min\{1, \sum_{j \in \{1,\ldots,n\} \setminus i} a_{y_i y_j}\}$. Another cost measure, which we will also use in this work, consider the cartographic preferences and is expressed by $c(S) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{y_i y_j})$.



	1	2	3	4	5	6	7	8	9	10	11	12
1	0.0	1.4	1.6	1.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	0.4	1.6	1.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	1.4	0.6	1.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	1.4	1.6	0.9	0.0	1.4	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	1.4	1.6	1.9	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	1.9	1.0	0.4	1.6	1.9	1.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	1.0	1.4	0.6	1.9	1.0	0.0	0.0	1.9
8	0.0	0.0	0.0	0.0	1.0	1.4	1.6	0.9	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	1.4	1.6	0.0	0.0	1.4	1.6	1.9
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.4	1.6	1.9
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.4	0.6	1.9
12	0.0	0.0	0.0	0.0	0.0	0.0	1.6	0.0	1.0	1.4	1.6	0.9

Figure 1: Example with $n = 3, S = \{4, 6, 9\}$, $f(S) = 3$ and $c(S) = 7$

Figure 1 illustrates an example with three points ($n = 3$) with four candidate positions each one ($y_1 \in \{1, 2, 3, 4\}, y_2 \in \{5, 6, 7, 8\}, y_3 \in \{9, 10, 11, 12\}$) and the corresponding matrix. Given the solution $S = (4, 6, 9)$ we note that $f(S) = \min(1, (a_4, a_6) + (a_4, a_9)) + \min(1, (a_6, a_4) + (a_6, a_9)) + \min(1, (a_9, a_4) + (a_9, a_6)) = \min(1, 1.4) + \min(1, 2.9) + \min(1, 1.4) = 3$ and $c(S) = (a_4, a_4) + (a_4, a_6) + (a_4, a_9) + (a_6, a_4) + (a_6, a_6) + (a_6, a_9) + (a_9, a_4) + (a_9, a_6) + (a_9, a_9) = 0.9 + 1.4 + 0.0 + 1.9 + 0.4 + 1.0 + 0.0 + 1.4 + 0.0 = 7$.

2.1 Tabu search based local search

For the local search procedure we used the basic ideas of the tabu search procedure proposed by [5]. Starting from a solution composed by the labels which corresponds the best position for each point $S = (y_1, y_2, \ldots, y_n)$, where $y_x = (x-1) * p + 1$ for $x = (1, \ldots, n)$, we investigate neighborhoods defined by moves which change the label of a point $x = (1, \ldots, n)$. We denote by $L_S(x)$ the label of point x in solution S. For each label $y_x \in \{(x-1){*}p+1, (x-1){*}p+2, \ldots, x{*}p\}$ we calculate the cost of this label in solution S' where $L_{S'}(x) = y_x$, $L_{S'}(\ell) = L_S(\ell)\ \forall \ell \neq x$ and we denote by $\Delta_S(y_x) = \sum_{j=1}^{n} (a_{y_x y_j})$ this value. A move $x(i \leftrightarrow k)$ is defined by changing the label of point x from i to k. The solution S' resulting from applying this move to solution S is characterized by $L_{S'}(x) = k$, $L_{S'}(\ell) = L_S(\ell)\ \forall \ell \neq x$. Whenever a move $x(i \leftrightarrow k)$ is performed, we forbid for a duration of parameter `TabuTenure` iterations all moves that would change the label of point x. We consider a candidate list with a number of parameter `CandidateListSize` labels (in solution) with higher $\Delta_S(y_x)$. At each tabu iteration, we scan the candidate list (in non-decreasing order of $\Delta_S(i)$) and we choose the not taboo move $x(i \leftrightarrow k)$ with the samaller value of $\Delta_S(k)$. We also consider the classical aspiration criteria which allows a taboo move to be selected if it improves the better solution so far.

We summarize in Figure 2 the procedure `TabuSearch`. Initializations are performed in lines 1-4, line 2 costs $O(v^2)$ and create an ordered list of size n in line 3, costs $O(n \log n)$. Iterations are performed along the loop in lines 4-36 until a solution with no overlaps is found or a total of `MaxTabuIt` tabu search iterations have been performed. The actualization of the parameters in line 7 cost $O(nv)$. At each iteration, the algorithm determines in loop 10-24 the best move. Scan the candidate list costs $O(v)$; verify, in line 12, if the point is in the tabu list can be done in constant time and verify, in line 15, if the move being examined would produce a solution with lower $c(S)$ has cost $O(v)$. The selected move is applied to the current solution in lines 27-35, dominated by line 29 with cost $O(vn)$. Each tabu iteration cost $O(vn)$ and the initialization step has cost $O(v^2)$.

Let parameter `NbLabelsOverlaps` $\leftarrow f(S)$ be the number of labels that has one or more overlaps in solution S. The tabu search procedure use the following parameters: (i) `TabuTenure` $\leftarrow$ `MinTabuLSize` + (`TabuFactor` $*$ `NbLabelsOverlaps`) where `MinTabuLSize` is the minimal size of the tabu list and `TabuFactor` is a weighted factor with relation to the number of labels that overlaps; (ii) `CandidateListSize` $\leftarrow \min(n,$ `MinCandidateLSize` $+$ `CandidateFactor` $*$ `NbLabelsOverlaps`$))$ where `MinCandidateLSize` is the minimal size of the candidate list and `CandidateFactor` is a weighted factor with relation to the number of labels that overlaps; (iii) `MaxTabuIt` is the maximum number of tabu search iterations. After some iterations the number of labels that overlaps decreases, and after each m consecutive iterations we recalculate parameters `TabuTenure` and `CandidateListSize`. If all the points are taboo, we increase the size of `CandidateFactor` by a factor `GrowingFactor` which, as a consequence, increases the size of the candidate list. After that, in each iteration this factor is divided by a factor `ReductionFactor` until it is equal to the original factor `CandidateBaseFactor`.

2.2 POPMUSIC

The basic POPMUSIC [4] frame can be summarized as follows:

```
procedure POPMUSIC($r, S$);
1  Solution $S$ composed of parts $s_1, \ldots, s_p$;
2  $O \leftarrow \emptyset$;
3  while $O \neq \{s_1, \ldots, s_p\}$ repeat
4      Select $s_i \notin O$;
5      Create a sub-problem $R_i$ composed of the $r$ parts most related to $s_i$;
6      Optimize $R_i$;
7      if $R_i$ has been improved then update $S$ and $O \leftarrow \backslash R_i$;
8      else $O \leftarrow O \cup \{s_i\}$;
end POPMUSIC.
```

```
procedure TabuSearch($n, p, A, S = (y_1, y_2, \ldots, y_n)$, MaxTabuIt);
1  iterations $\leftarrow 0$; TabuList $\leftarrow \emptyset$;
2  Calculate $\Delta_S(i)\ \forall i = \{1, \ldots, (n * p)\}$;
3  CandidateList $\leftarrow (y_{k_1}, y_{k_2}, \ldots, y_{k_{\texttt{CandidateListSize}}}) \subseteq S$
             where $\Delta_S(y_{k_1}) \geq \Delta_S(y_{k_2}) \geq \ldots \geq \Delta_S(y_{k_{\texttt{CandidateListSize}}})$;
4  BestSolution $\leftarrow S$;
5  while (iterations $<$ MaxTabuIt) and $c$(BestSolution) $> 0$ do
6      if (iterations $\% m == 0$)
7          Actualize TabuListSize and CandidateListSize;
8      end if;
9      kRetained $\leftarrow \infty$; iRetained $\leftarrow \infty$; MinDelta $\leftarrow \infty$;
10     forall $y_i \in$ CandidateList do
11         forall $k \in \{(i-1)*p+1, (i-1)*p+2, \ldots, i*p\} \setminus y_i$ do
12             Autorized $\leftarrow (i \notin$ Tabulist$)$;
13             if ($\Delta_S(k) <$ MinDelta)
14                 if not (Autorized) then do
15                     $S' \leftarrow S$; $L_{S'}(i) \leftarrow k$;
16                     Aspired $\leftarrow (c(S') <$ BestCost$)$;
17                 end if;
18                 if (Autorized or Aspired) then do
19                     kRetained $\leftarrow k$; iRetained $\leftarrow i$;
20                     MinDelta $\leftarrow \Delta_S(k)$;
21                 end if;
22             end if;
23         end forall;
24     end forall;
25     iterations $\leftarrow$ iterations $+ 1$;
26     if (kRetained $= \infty$) Actualize CandidateListSize;
27     else do
28         TabuList $\leftarrow$ TabuList $\cup$ yRetained for the next TabuTenure iterations;
29         Update $\Delta_S(i)\ \forall i = \{1, \ldots, (n * p)\}$;
30         Reorder CandidateList;
31         $L_S$(iRetained) $\leftarrow$ kRetained;
32         if $c(S) < c$(BestSolution) then do
33             BestSolution $\leftarrow S$;
34         end if;
35     end if;
36 end while;
37 return BestSolution;
end TabuSearch.
```

Figure 2: Pseudo-code of tabu search based local search procedure for PFLP

The set O of part corresponds to seed parts that have been used to define sub-problems that have been unsuccessfully optimized. Once O contains all the parts of the complete solution, then all sub-problems have been examined without success and the process stops. It has one parameter, r that controls the size of the sub-problem to be optimized.

We now define the choices for POPMUSIC elements used in our implementation. Each point n defines a part $s_1, \ldots, s_n$, and consequently each part is composed by p labels. The next *seed* part is considered in index order $1, \ldots, n$. Problem R_i will be created as follows: let L be a list of points, each one with an associated cost. Initially L is empty. For each label of the point being inserted, scan the labels that overlap with it. For each one of them verify if its correspondent point is already in L. If so, increment the cost of this point by one, otherwise include this point with cost 1 in L. The next part most related to the *seed* part s_i will be the point from L with the higher cost. Repeat this procedure until the sub-problem is composed by r parts or there is no more part which has conflict with the sub-problem to be inserted in R_i, i.e., L is empty.

The pseudo-code of our POPMUSIC approach for PFLP is given in Figure 3. Initializations are performed in lines 1-3. A seed part not in O is selected in line 5 and the corresponding sub-problem with r parts is constructed in lines 6-8. In line 9 we apply the tabu search based heuristic (described in the previous subsection) to the current solution $\overline{S}$ of the sub-problem being examined. If the number of labels of the sub-problem with conflicts decreases, then we update the solution of the original problem S in line 11. Next, we test if this change has improved the best solution and, if so, we update the best solution in line 13 and, in line 14, include R_i in O (this is faster than just making O empty). Otherwise, in line 15, we include in set O the current seed part. The loop in lines 4-17 stops when we find an optimal solution ($f(S) = 0$) or when there is no more seed part to improve.

3 Preliminary Computational Results

All computational experiments were performed on a Pentium 4, 2.8 GHz with 256 MB of RAM memory. Algorithm POPMUSIC_PFLP was coded in `C` and compiled with version 3.2.2 of the `gcc` compiler with the optimization flag -O3. We considered the set of test problems introduced by L.A.N. Lorena and available from `http://www.lac.inpe.br/~lorena/instancias.html`. There are twenty five instances for each value of the number of points $n \in \{25, 100, 250, 500, 750, 1000\}$ with four potential label positions for each point. We have considered only instances with $n \geq 250$ in our computational results, since the other instances are very easy to solve.

The implementation assumed the following values for the tabu search parameters [2]: `MinTabuLSize` equal to 9; `TabuFactor` equal to 0.77; `MinCandidateLSize` equal to 18; `CandidateBaseFactor` equal to 0.73; `GrowingFactor` equal to 15; `ReductionFactor` equal to 1.3; m equal to 50 and `MaxTabuIt` $\in \{50n, 110n, 500n\}$.

To investigate the effectiveness of using POPMUSIC strategy, we compare our algorithm with a basic tabu search presented in this paper. We try three different settings with both algorithms: For POPMUSIC we try $r = 10$, $r = 30$ and $r = 70$ (and `iterTabu` $= 10 * r$ for the optimization process embeds in POPMUSIC). For tabu search, we try `MaxTabuIt` $= 50 * n$, $110 * n$ and $200 * n$. Table 1 summarizes these results. We provide the following statistics

```
procedure POPMUSIC_PFLP($n, p, r$, iterTabu, $A, S = (y_1, y_2, \ldots, y_n)$);
1  Solution $S$ composed of parts $s_1, \ldots, s_n$;
2  $O \leftarrow \emptyset$;
3  BestSolution $\leftarrow S$;
4  while $O \neq \{s_1, \ldots, s_n\}$ and $f$(BestSolution) $> 0$ do repeat
5      Select $s_i \notin O$;
6      Create sub-problem $R_i$ composed of the $r$ parts most related to $s_i$ from $S$;
7      Construct the correspondent $r \times r$ overlap matrix $B$ for sub-problem $R_i$;
8      Construct the correspondent solution $\overline{S} = (\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_r)$;
9      $S' \leftarrow$ TabuSearch($r, p, B, \overline{S}$, iterTabu);
10     if $f(S') < f(\overline{S})$ then do
11         Transform solution $S'$ with $r$ points in solution $S$ with $n$ points;
12         if $f(S) < f$(BestSolution) then do
13             BestSolution $\leftarrow S$;
14             $O \leftarrow O \setminus R_i$;
15         else $O \leftarrow O \cup \{s_i\}$;
16     end if;
17 end while;
18 return BestSolution;
end POPMUSIC_PFLP.
```

Figure 3: Pseudo-code of POPMUSIC based procedure for PFLP

average over the 25 instances, either for objectives without preferences ($w(y_x) = 0$) or for objectives with preferences ($w(y_x) \in \{0.0, 0.4, 0.6, 0.9\}$): % is the percentage of label placed without conflict, $c(S)$, $f(S)$ and CPU time.

We also compare, in Table 1, the heuristic POPMUSIC_PFLP with the Genetic Heuristic of Yamamoto et al. [6] (column CGA(best) reports the average result for six trials and CGA(average) is the best result) and the Tabu search Heuristic of Yamamoto et al [5] (column Tabu). Let us mention that the numerical results provided in these last references only consider the % of labels placed without conflicts. The processor used by the authors of Tabu [5] was a Sun Sparc 20 workstation (therefore a computer about 45 times slower than ours, according to Dongarra's factor). The computational effort corresponding to the results for Tabu [5] is therefore similar to those of our tabu search when run between $110n$ and $500n$ iterations. The processor used for CGA [6] was a Pentium III (at least 550 MHz ?), therefore a computer about 10 times slower than ours). These authors only provides the computational time to reach the best solution. Therefore, the computational times are difficult to compare, but it is sure that the computational effort for our methods was significantly lower than those of CGA.

First, we notice that our tabu search implementation provides solutions of similar quality than those reported by the Tabu search heuristic of Yamamoto et al. [5], which is coherent. Then, we notice that the tabu search is not able to find solutions as good as POPMUSIC ones, even if it runs for much longer CPU times (there are however three exceptions for the % of label placed without conflicts, but the other objectives are much higher). Therefore, the POPMUSIC approach was fundamental to find better solutions. Strangely, increasing the sub-problem size in POPMUSIC seems to be counter-productive: the computational time are higher and the solution worse. This perhaps translates the fact the tabu search embedded in POPMUSIC frame as optimization procedure is not able to find good solutions to problems with a moderately large number of labels.

We propose a new heuristic for solving the point feature label placement problem based on the application of POPMUSIC methodology. Preliminary results showed that the proposed heuristic outperformed other recent metaheuristic approaches in the literature, in terms of solution quality. To obtain solutions of better quality than those of this preliminary study, extensions are under development.

References

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

[2] Burri, G., Taillard, Éric. (2003): "Problème du placement géographique de labels". In: *Raport de travail de diplôme*, École d'ingénieurs du canton de Vaud.

[3] Taillard, Éric, and Burri, G. (2004): "POPMUSIC pour le placement de légende sur de plans". In: *Francoro IV*, 95-97.

[4] Taillard, Éric, and Voss, S. (2001): "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.

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

[6] Yamamoto, M., and Lorena, L.A.N. (2003): "A Constructive Genetic Approach to Point-Feature Cartographic Label Placement". Presented at *The Fifth Metaheuristics International Conference 2003 (MIC2003)*, Kyoto, Japan, August, 2003.

Table 1: Computational results

	without preferences				with preferences			
	250							
	%	$c(S)$	$f(S)$	s	%	$c(S)$	$f(S)$	s
PopMusic(10)	100.00	0.0	0.0	0.00	99.97	0.10	0.08	0.00
PopMusic(30)	100.00	0.0	0.0	0.00	99.94	0.18	0.16	0.00
PopMusic(70)	100.00	0.0	0.0	0.00	99.94	0.18	0.16	0.00
Tabu(50n)	99.84	0.4	0.4	0.01	99.42	19.02	1.44	0.02
Tabu(110n)	99.84	0.4	0.4	0.02	99.42	19.02	1.44	0.07
Tabu(500n)	99.84	0.4	0.4	0.12	99.42	19.02	1.44	0.32
CGA(best) [6]	100.00	-	-	0.6[a]	-	-	-	-
CGA(average) [6]	100.00	-	-	0.6[a]	-	-	-	-
Tabu [5]	100.00	-	-	28.0[b]	-	-	-	-
	500							
	%	$c(S)$	$f(S)$	s	%	$c(S)$	$f(S)$	s
PopMusic(10)	99.60	2.00	2.00	0.02	99.34	4.13	3.32	0.03
PopMusic(30)	99.66	1.68	1.68	0.03	99.14	4.67	4.28	0.08
PopMusic(70)	99.66	1.68	1.68	0.03	99.05	5.13	4.76	0.14
Tabu(50n)	99.60	2.00	2.00	0.28	97.90	79.68	10.52	0.72
Tabu(110n)	99.60	2.00	2.00	0.43	99.08	79.28	9.6	2.10
Tabu(500n)	99.62	1.92	1.92	2.75	98.25	79.11	8.76	8.47
CGA(best) [6]	99.60	-	-	21.5[a]	-	-	-	-
CGA(average) [6]	99.60	-	-	21.5[a]	-	-	-	-
Tabu [5]	99.20	-	-	114.0[b]	-	-	-	-
	750							
	%	$c(S)$	$f(S)$	s	%	$c(S)$	$f(S)$	s
PopMusic(10)	96.26	30.64	28.04	0.12	95.18	52.17	36.16	0.15
PopMusic(30)	97.52	19.04	18.56	0.41	96.80	30.27	24.00	0.94
PopMusic(70)	97.63	18.08	17.76	1.33	96.04	36.60	29.68	4.17
Tabu(50n)	96.72	24.88	24.60	1.37	94.74	206.60	39.48	4.12
Tabu(110n)	96.79	24.40	24.08	2.29	94.73	206.60	39.48	9.18
Tabu(500n)	96.95	23.36	22.84	14.21	95.16	205.81	36.32	41.99
CGA(best) [6]	97.10	-	-	228.9[a]	-	-	-	-
CGA(average) [6]	96.80	-	-	195.9[a]	-	-	-	-
Tabu [5]	96.80	-	-	245.0[b]	-	-	-	-
	1000							
	%	$c(S)$	$f(S)$	s	%	$c(S)$	$f(S)$	s
PopMusic(10)	86.71	153.76	132.92	0.39	84.22	254.39	157.80	0.47
PopMusic(30)	91.30	92.56	87.00	1.92	89.18	150.97	108.24	2.92
PopMusic(70)	92.16	83.12	78.40	7.57	88.57	152.35	114.32	12.87
Tabu(50n)	90.08	101.76	99.20	3.52	88.18	440.07	118.20	8.33
Tabu(110n)	90.19	100.80	98.12	7.00	88.84	437.73	111.60	18.42
Tabu(500n)	90.45	98.40	95.48	35.14	89.13	436.22	108.68	81.53
CGA(best) [6]	90.70	-	-	1227.2[a]	-	-	-	-
CGA(average) [6]	90.40	-	-	981.8[a]	-	-	-	-
Tabu [5]	90.00	-	-	1179.0[b]	-	-	-	-

[a] time to reach the best solutions on a Pentium III.
[b] time to reach the best solutions on a Sun Sparc 20.
'-' is used to indicate information not available.