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 [2]. Independent of the features being labeled these problems are NP-hard.

POPMUSIC is a general optimization method especially designed for optimizing the solutions of large instances of combinatorial problems and can be seen as an LNS (Large Scale Neighbourhood, see [5]). 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 [1, 4] 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 $v = n \times p$ 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 $y_x \in \{ (x - 1) \times p + 1, (x - 1) \times p + 2, (x - 1) \times p + 3, \ldots, x \times p \}$. Associated with each label $y_x$ there is a weight $w(y_x) \in \{0, 0.4, 0.6, 0.9\}$ (weights are arbitrary and need to be $< 1$) which corresponds to the quality of its placement, lower values indicating best positions [6]. 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 $f(S) = \sum_{i=1}^{n} \min\{1, \sum_{j\in\{1,\ldots,n\}\setminus i} a_{yi,yj}\}$. Another cost measure, which we will also use in this work, consider the cartographic preferences and is expressed by $c(S) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{yi,yj}$.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>1.4</td>
<td>1.6</td>
<td>1.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.4</td>
<td>1.6</td>
<td>1.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>1.0</td>
<td>0.6</td>
<td>1.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>1.4</td>
<td>1.6</td>
<td>0.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.4</td>
<td>1.6</td>
<td>1.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.4</td>
<td>1.6</td>
<td>1.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.4</td>
<td>0.6</td>
<td>1.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.4</td>
<td>1.6</td>
<td>0.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.4</td>
<td>1.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.4</td>
</tr>
<tr>
<td>10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>11</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>12</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.4</td>
<td>1.6</td>
<td>0.0</td>
<td>1.0</td>
<td>1.4</td>
<td>1.6</td>
</tr>
</tbody>
</table>

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

2.1 Tabu search based local search

For the local search procedure we used the basic ideas of the tabu search procedure proposed by [6]. 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 $y_x = (x - 1) \times p + 1$ 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 $y_x \in \{ (x - 1) \times p + 1, (x - 1) \times p + 2, \ldots, x \times p \}$ 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 $\text{TabuTenure}$ iterations all moves that would change the label of point $x$. We consider a candidate list with a number of parameter $\text{CandidateListSize}$ labels (in solution) with higher $\Delta_S(y_x)$. At each tabu iteration, we scan the candidate list (in non-decreasing order of $\Delta_S(y_x)$) and we choose the not taboo move $x(i \leftrightarrow k)$ with the smaller 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. The tabu search procedure stops when a solution with no overlaps is found or a total of $\text{MaxTabuIt}$ tabu search iterations have been performed.

2.2 POPMUSIC

The basic POPMUSIC [5] frame can be summarized as follows:

\begin{verbatim}
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 $O \leftarrow O \setminus R_i$;
8       else $O \leftarrow O \cup \{s_i\}$;
9 end POPMUSIC.
\end{verbatim}

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. POPMUSIC frame doesn’t formally define the concept of relation between parts. This definition must be done by the designer of a popmusic-based method (e.g. similar to the definition of the concept of tabu list in tabu search).

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 2. Initializations
procedure POPMUSIC_PFLP(n, p, r, iterTabu, A, S = (y_1, y_2, ..., y_n));
1 Solution S composed of parts s_1, ..., s_n; O ← ∅; BestSolution ← S;
2 while O ≠ {s_1, ..., s_n} and f(BestSolution) > 0 repeat
3 Select s_i /∈ O;
4 Create sub-problem R_i composed of the r parts most related to s_i from S;
5 Construct the correspondent solution S = (y_1, y_2, ..., y_r);
6 Apply optimization process Tabu Search on solution S producing solution S';
7 if f(S') < f(S) then do
8 Update the position of the r points of S' in the whole solution S;
9 BestSolution ← S; O ← O \ R_i;
10 else O ← O ∪ {s_i};
11 return BestSolution;
end POPMUSIC_PFLP.

Figure 2: Pseudo-code of POPMUSIC based procedure for PFLP

are performed in line 1. A seed part not in O is selected in line 3 and the corresponding sub-problem with r parts is constructed in lines 4-5. In line 6 we apply the tabu search based heuristic (described in the previous subsection) to the current solution 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 8, update the best solution in line 9 and remove R_i from O. Otherwise, in line 10, we include in set O the current seed part. The loop in lines 2-10 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 ∈ {25, 100, 250, 500, 750, 1000} with four potential label positions for each point. We have considered only instances with n ≥ 250 in our computational results, since the other instances are very easy to solve.

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 MaxTabuIt = 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 average over the 25 instances, either for objectives without preferences (w(y_x) = 0) or for objectives with preferences (w(y_x) ∈ {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. [7] (column CGA(best)

Vienna, Austria, August 22–26, 2005
reports the average result for six trials and CGA(average) is the best result) and the Tabu search Heuristic of Yamamoto et al [6] (column Tabu). Let us mention that the numerical results provided in these last references only consider the % of labels placed without conflicts. The computational times indicated in Table 1 for Tabu [6] and CGA [7] have been divided by Dongarra’s factor [3] in order to have comparable times. We note that the computational effort corresponding to the results for Tabu [6] is therefore similar to those of our tabu search when run between 110n and 500n iterations. Yamamoto et al. [7] only provide 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. [6], 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 few 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. To conclude, in Table 1, we see that POPMUSIC approach is much faster than the best methods previously published while providing solutions of higher quality.

References


Vienna, Austria, August 22–26, 2005
## Table 1: Computational results

<table>
<thead>
<tr>
<th>without preferences</th>
<th>250</th>
<th>with preferences</th>
<th>500</th>
<th>750</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%</td>
<td>c(S)</td>
<td>f(S)</td>
<td>s</td>
<td>%</td>
</tr>
<tr>
<td>PopMusic(10)</td>
<td>100.00</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>99.97</td>
</tr>
<tr>
<td>PopMusic(30)</td>
<td>100.00</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>99.94</td>
</tr>
<tr>
<td>PopMusic(70)</td>
<td>100.00</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>99.94</td>
</tr>
<tr>
<td>Tabu(50n)</td>
<td>99.84</td>
<td>0.4</td>
<td>0.4</td>
<td>0.01</td>
<td>99.42</td>
</tr>
<tr>
<td>Tabu(110n)</td>
<td>99.84</td>
<td>0.4</td>
<td>0.4</td>
<td>0.02</td>
<td>99.42</td>
</tr>
<tr>
<td>Tabu(500n)</td>
<td>99.84</td>
<td>0.4</td>
<td>0.4</td>
<td>0.12</td>
<td>99.42</td>
</tr>
<tr>
<td>CGA(best) [7]</td>
<td>100.00</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CGA(average) [7]</td>
<td>100.00</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Tabu [6]</td>
<td>100.00</td>
<td>-</td>
<td>-</td>
<td>0.06a</td>
<td>-</td>
</tr>
</tbody>
</table>

| PopMusic(10)        | 99.62 | 1.92 | 1.92 | 0.04 | 99.41 | 3.36 | 2.96 | 0.05 |
| PopMusic(30)        | 99.66 | 1.68 | 1.68 | 0.03 | 99.15 | 4.66 | 4.24 | 0.06 |
| PopMusic(70)        | 99.66 | 1.68 | 1.68 | 0.04 | 99.06 | 5.07 | 4.68 | 0.12 |
| 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) [7]       | 99.60 | -    | -    | 2.15a | -    | -    | -    | -   |
| CGA(average) [7]    | 99.60 | -    | -    | 2.15a | -    | -    | -    | -   |
| Tabu [6]            | 99.20 | -    | -    | 2.53b | -    | -    | -    | -   |

| PopMusic(10)        | 97.02 | 23.04 | 22.36 | 0.31 | 96.47 | 33.17 | 26.48 | 0.42 |
| PopMusic(30)        | 97.62 | 18.08 | 17.88 | 0.54 | 96.65 | 30.56 | 25.12 | 0.89 |
| PopMusic(70)        | 97.64 | 17.84 | 17.68 | 1.50 | 95.96 | 36.67 | 30.28 | 3.25 |
| 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) [7]       | 97.10 | -    | -    | 22.89a | -    | -    | -    | -   |
| CGA(average) [7]    | 96.80 | -    | -    | 19.59a | -    | -    | -    | -   |
| Tabu [6]            | 96.80 | -    | -    | 5.44b | -    | -    | -    | -   |

| PopMusic(10)        | 89.82 | 110.00 | 101.76 | 1.23 | 88.57 | 149.48 | 114.32 | 1.35 |
| PopMusic(30)        | 91.99 | 82.56 | 80.12 | 3.15 | 89.15 | 138.95 | 108.52 | 4.59 |
| PopMusic(70)        | 92.18 | 80.16 | 78.16 | 9.05 | 87.69 | 155.08 | 123.08 | 13.95 |
| 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) [7]       | 90.70 | -    | -    | 122.72a | -    | -    | -    | -   |
| CGA(average) [7]    | 90.40 | -    | -    | 98.18a | -    | -    | -    | -   |
| Tabu [6]            | 90.00 | -    | -    | 26.2b | -    | -    | -    | -   |

*a* time to reach the best solutions on a Pentium III divided by 10 [3].

*b* time to reach the best solutions on a Sun Sparc 20 divided by 45 [3].

'-' is used to indicate information not available.