A New Evaluation Function for the MinLA Problem

Eduardo Rodriguez-Tello* Jin-Kao Hao* Jose Torres-Jimenez†

*LERIA, Université d'Angers
2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
{ertello, hao}@info.univ-angers.fr

†Mathematics Department, University of Guerrero
54 Carlos E. Adame, 39650 Acapulco Guerrero, Mexico
jtj@uagro.mx

1 Introduction

The *minimum linear arrangement problem* (MinLA) was first stated by Harper in [5]. His aim was to design error-correcting codes with minimal average absolute errors on certain classes of graphs. Latter, in the 1970's it was used as an abstract model of the placement phase in VLSI layout, where nodes of the graph represented modules and edges represented interconnections. In this case, the cost of the arrangement measures the total wire length [1]. MinLA arises also in other application areas like graph drawing, software diagram layout and job scheduling [3].

The MinLA problem can be defined formally as follows. Let $G(V,E)$ be a finite undirected graph, where V ($|V| = n$) defines the set of vertices and $E \subseteq V \times V = \{\{i,j\} \mid i,j \in V\}$ is the set of edges. Given a one-to-one function $\varphi : V \rightarrow \{1..n\}$, called a linear arrangement, the total edge length for G with respect to arrangement φ is defined by:

$$LA(G,\varphi) = \sum_{(u,v)\in E} |\varphi(u) - \varphi(v)| \tag{1}$$

Then the MinLA problem consists in finding an arrangement φ for a given G so that $LA(G,\varphi)$ is minimized.

As is the case with many graph layout problems, finding the minimum linear arrangement is NP-hard and the corresponding decision problem is NP-complete [4]. Only in very special cases it is possible to find the optimal arrangement in polynomial time (see [3] for a detailed survey). Nowadays, the best polynomial time approximation algorithm for MinLA gives a $O(log\ n)$ approximation factor for general graphs [8]. However, this algorithm presents the disadvantage of having to solve a linear program with an exponential number of constraints, making it impractical for large graphs.

As an indispensable alternative, several heuristic methods have been proposed. Some examples are: a binary balanced decomposition tree heuristic [2] and a multi-scale algorithm

[6]. In [7], Petit presents several heuristics for MinLa that belong to the families of Greedy algorithms, local search algorithms (Hillclimbing, Metropolis and Simulated Annealing) and Spectral Sequencing. He concluded that the heuristic that achieved the best results was Simulated Annealing.

All these algorithms have a point in common, all of them evaluate the quality of a solution (linear arrangement) as the change in the objective function $LA(G, \varphi)$. It represents a potential drawback because different linear arrangements can result in the same total edge length. This incomplete information can prevents the search process from finding better solutions.

Taking this idea in mind, and given that one of the most important elements in heuristic search is how the quality of a solution is evaluated, a new evaluation function (namely Ψ) is proposed in this paper. This new evaluation function is able to capture even the smallest improvement that orients the searching of better solutions and permits to distinguish arrangements with the same total edge length.

The paper is organized as follows: In Section 2 the new evaluation function Ψ is introduced. Section 3 shows a Simulated Annealing algorithm which is used to compare Ψ and the classical evaluation function LA. Section 4 presents an experimental comparison between Ψ and LA. Finally, in Section 5 some conclusions are presented.

2 The Ψ evaluation function

The choice of the evaluation function is an important aspect of any search procedure. Firstly, in order to efficiently test each potential solution, the evaluation function must be as simple as possible. Secondly, it must be sensitive enough to locate promising search regions on the space of solutions. Finally, the evaluation function must be consistent: a solution that is better than others must return a better value.

The classical evaluation function (LA) does not fulfill these requirements, because using it different linear arrangements can result in the same total edge length. Given this negative feature we have developed a new evaluation function, called Ψ, which permits to better discriminate arrangements with the same total edge length. The proposed evaluation function not only measures the total edge length for G with respect to arrangement φ, it also evaluates how much a linear arrangement can improve. Next, we present some preliminary concepts used in its definition.

Consider the contribution $L(u, \varphi)$ of a vertex u with respect to the linear arrangement φ defined by: $L(u, \varphi) = \sum_{v \in A(u)} |\varphi(u) - \varphi(v)|$, where $A(u)$ is the set of adjacent vertices to u.

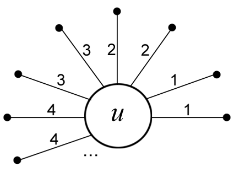

Figure 1: The best possible arrangement for a node u.

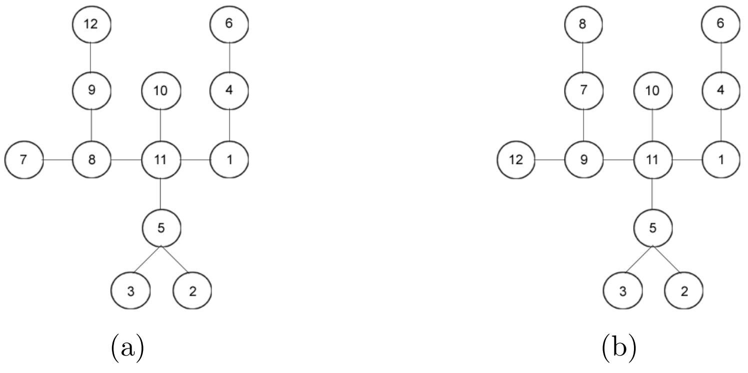

Figure 2: (a) Arrangement φ with $LA = 35$. (b) Arrangement φ' with $LA = 35$.

Remark that in the best possible arrangement, a node must have two incident edges that contribute 1, two incident edges that contribute 2, two incident edges that contribute 3, and so on (see Figure 1). Therefore, the maximal contribution $\bar{L}(u, \varphi)$ of a node u with degree d can be defined as follows:

$$\bar{L}(u,\varphi) = \begin{cases} \frac{d+1}{2} + \sum_{i=1}^{\frac{d-1}{2}} 2i = \frac{(d+1)^2}{4}, & \text{if } d \text{ is odd} \\ \sum_{i=1}^{\frac{d}{2}} 2i = \frac{d(d+2)}{4}, & \text{if } d \text{ is even} \end{cases} \tag{2}$$

Using this bound we are able to compute the potential improvement of an arrangement φ using the expression $I(G, \varphi) = \sum_{u \in V}[L(u, \varphi) - \bar{L}(u, \varphi)]$. Additionally, the potential improvement can be normalized ($0 < I_{norm}(G, \varphi) < 1$) by using the formula $I_{norm}(G, \varphi) = 1 - 1/I(G, \varphi)$. Then, we can define the new Ψ evaluation function as follows:

$$\Psi(G, \varphi) = LA(G, \varphi) + I_{norm}(G, \varphi) \tag{3}$$

To illustrate the computation of this new evaluation function, let us consider the graph in Fig. 2(a). For this particular graph $n = 12$, $LA(G, \varphi) = 35$ and $I(G, \varphi) = 40$.

Then, by making the substitution of these values in the Formula 3 we obtain: $\Psi(G, \varphi) = 35 + (1 - \frac{1}{40}) = 35.975$. In contrast if Ψ is computed for the arrangement φ' showed in Fig. 2(b) a different and smaller value is obtained: $\Psi(G, \varphi') = 35 + (1 - \frac{1}{38}) = 35.973$. It means that the arrangement φ is better than φ'. Indeed it is better because its potential improvement is higher.

In this sense this new evaluation function is much more discriminating than LA and leads to smoother landscapes during the search process. This is possible because Ψ allows to capture even the smallest improvement that orients the searching process of solutions.

3 A SA algorithm for solving the MinLA problem

To evaluate the practical usefulness of the Ψ evaluation function, a Simulated Annealing (SA) algorithm was developed. Next some details of the implementation proposed are presented:

Evaluation Function. In this implementation we have included the Ψ evaluation function whose formal definition is presented in Formula 3.

Neighborhood Function. The neighborhood of a solution $N(\varphi)$ in our implementation contains all the arrangements φ' obtained by swapping two randomly selected vertices of the current arrangement φ.

Initial Solution. The initial solution is the starting configuration used as input for the algorithm. This SA implementation generates randomly the initial arrangement.

Cooling Schedule. In this SA algorithm the proportional cooling schedule is used ($T_n = T_{n-1} * 0.95$). The initial temperature was fixed at 10 and the final temperature (T_f) at 0.2.

Termination Condition. The algorithm stops either if the current temperature reaches T_f, or if the number of accepted configurations at each temperature falls below the limit of 25. The maximum number of accepted configurations at each temperature ($maxConfigurations$), depends directly on the number of nodes in the graph (n), because more moves are required for larger graphs ($maxConfigurations = 30n^{3/2}$).

4 Computational experiments

In this section, we present the experiments accomplished to evaluate the performance of Ψ over a set of 21 benchmark instances[1]. For these experiments the above SA algorithm is used. The code, programmed in C, was compiled with *gcc* using the optimization flag -*O*3 and ran into a cluster of 5 nodes, each having a Xeon bi-CPU at 2 GHz and 1 GB of RAM with Linux. Due to the non-deterministic nature of th SA algorithm, 20 independent runs were executed for each of the selected benchmark instances. All the results reported here are data averaged over the 20 corresponding runs.

The set of problem instances is the same proposed by Petit [7] and used in [2, 6]. It consists of six different families: Uniform random graphs (randomA* class), Geometric random graphs (randomG* class), Graphs with known optima (trees, hypercubes and meshes), Graphs from finite element discretizations (3elt, airfoil1 and whitaker3), Graphs from VLSI design (c*y class) and Graphs from graph-drawing competitions (gd* class). All the graphs included in this test-suite have 1000 vertices or more, except for some instances in the gd* class.

The criteria used for evaluating the performance of Ψ are the same as those used in the literature: the average total edge length for each instance and the average CPU time in seconds.

[1]These instances are available at: http://www.lsi.upc.es/ jpetit/MinLA/Experiments/jpetit-extra.tar.gz

4.1 Comparison between Ψ and LA

In order to compare both evaluation functions we used them within the SA algorithm presented in Section 3 (call these SA algorithms SA-Ψ and SA-LA) and test them on the set of 21 instances. Both SA-Ψ and SA-LA were run 20 times on each instance and the results are presented in Table 1. In this table columns 1 to 3 show the name of the graph, the number of vertices and edges. Columns 4 to 7 shows the average total edge length and the average CPU time in seconds for the 20 runs of the SA algorithm that uses the Ψ and the LA evaluation function respectively. The last column presents the improvement obtained when the Ψ evaluation function was used.

The results presented in Table 1 show clearly that the new evaluation function Ψ allows the SA algorithm to obtain better results for many classes of graphs with very weak additional computing time. We can observe an average improvement of 9.18%, with a peak of 62.31% (see column Improvement). So the superiority of Ψ over LA is confirmed.

Table 1: Results obtained with two SA algorithms using Ψ and LA.

			SA-Ψ		SA-LA		%
Graph	n	Edges	Cost	Time	Cost	Time	Improvement
randomA1	1000	4974	914523.80	88.99	934880.00	74.50	2.18
randomA2	1000	24738	6665621.10	223.65	6714392.00	182.93	0.73
randomA3	1000	49820	14376963.90	356.07	14445089.00	289.66	0.47
randomA4	1000	8177	1786088.90	105.72	1810207.00	91.19	1.33
randomG4	1000	8173	229349.70	81.29	277593.00	87.10	17.38
bintree10	1023	1022	4274.40	42.11	11341.00	55.10	62.31
hc10	1024	5120	523776.00	92.30	528829.00	90.90	0.96
mesh33x33	1089	2112	38395.50	56.52	44430.00	89.80	13.58
3elt	4720	13722	427555.60	785.71	481815.00	756.60	11.26
airfoil1	4253	12289	347672.80	693.64	384013.00	665.80	9.46
whitaker3	9800	28989	1216866.30	3273.27	1231912.00	3230.60	1.22
c1y	828	1749	64490.80	36.96	70710.00	46.51	8.80
c2y	980	2102	82120.10	48.64	90158.00	54.25	8.92
c3y	1327	2844	139480.20	81.35	151622.00	87.15	8.01
c4y	1366	2915	118541.10	87.83	131106.00	91.30	9.58
c5y	1202	2557	106568.00	68.03	118541.00	83.90	10.10
gd95c	62	144	507.30	0.39	525.00	0.67	3.37
gd96a	1096	1676	105722.20	57.73	124704.00	66.76	15.22
gd96b	111	193	1420.90	1.60	1437.00	1.08	1.12
gd96c	65	125	520.50	0.45	531.00	0.56	1.98
gd96d	180	228	2402.10	2.62	2523.00	2.92	4.79
Average			1292993.39	294.52	1312207.52	288.06	9.18

5 Conclusion

In this paper, we have introduced the Ψ evaluation function for the MinLA problem. It allows to better orient the search process by producing a smoother landscape.

To validate the practical usefulness of Ψ, two versions of a basic SA (SA-Ψ and SA-LA) were implemented. They were compared using a set of well known benchmarks and the results showed that for many classes of graphs an average improvement of 9.18% can be achieved when Ψ is used.

Finally, let us notice that the Ψ evaluation function proposed in this paper can be used by other metaheuristic algorithms (Genetic Algorithms, Tabu Search, Scatter Search) to boost their performance.

More generally, we think that research of new evaluation function for combinatorial problems is a very important issue and worthy of more attention of researchers. Indeed, it is this function that guides the exploration and exploitation process of a heuristic algorithm.

Acknowledgments. This work was supported in part by the Franco-Mexicain Computer Science Laboratory (LAFMI), the CONACyT Mexico and the CPER COM project.

References

[1] Adolphson, D., and Hu, T. (1973): "Optimal Linear Ordering". In: *SIAM J. Appl. Math.* **25** (3), 403–423.

[2] Bar-Yehuda, R., Even, G., Feldman, J., and Naor, S. (2001) "Computing an Optimal Orientation of a Balanced Decomposition Tree for Linear Arrangement Problems". In: *Journal of Graph Algorithms and Applications* **5** (4), 1–27.

[3] Diaz, J., Petit, J., and Serna, M. (2002): "A Survey of Graph Layout Problems". In: *ACM Comput. Surv.* **34** (3), 313–356.

[4] Garey, M., and Johnson, D. (1979): *Computers and Intractability: A guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, USA.

[5] Harper, L. (1964): "Optimal Assignment of Numbers to Vertices". In: *Journal of SIAM* **12** (1), 131–135.

[6] Koren, Y., and Harel, D. (2002): "A Multi-scale Algorithm for the Linear Arrangement Problem". In: *Lecture Notes in Computer Science* **2573**, 293–306.

[7] Petit, J. (1998): "Approximation Heuristics and Benchmarkings for the MinLA Problem". In: *Algorithms and Experiments (ALEX98)*, pp. 112–128.

[8] Rao, S., and Richa, A. (1998): "New Approximation Techniques for Some Ordering Problems". In: *Proc. of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 211–218.