GRASP with Path Relinking for the Constrained Incremental Graph Drawing Problem

A. Martínez-Gavara
Dept. Estadística i Inv. Operativa
Universitat de València
València, España
gavara@uv.es

A. Napoletano
Dept. Mathematics and Applications
University of Napoli Federico II
Napoli, Italy
antonio.napoletano2@unina.it

P. Festa
Dept. Mathematics and Applications
University of Napoli Federico II
Napoli, Italy
paola.festa@unina.it

T. Pastore
Dept. Mathematics and Applications
University of Napoli Federico II
Napoli, Italy
tommaso.pastore@unina.it

R. Martí
Dept. Estadśtica i Inv. Operativa
Universitat de València
València, España
rafael.marti@uv.es

***Abstract*—Graph Drawing is a well-established area in Computer Science, with applications from scheduling to software diagrams. The main quality desired for drawings is readability, and edge crossing minimization is a well-recognized method for a good representation of them. This work focuses in incremental graph drawing problems, with the aim to minimize the number of edge crossings while satisfying some constraints to preserve the absolute position of the vertices in previous drawings. We propose a mathematical model and a GRASP (Greedy Randomized Adaptive Search Procedure) with PR (Path Relinking) methodology to solve this problem. Finally, we compare our methodology with `CPLEX` and heuristic solutions obtained by a well-known black-box solver, `LocalSolver`.**

***Index Terms*—heuristics, graph drawing, path relinking**

I. Introduction

Graph Drawing is a relevant topic of research in areas such as workflow visualization, database modeling, bioinfomatics and decision diagrams. The difficulty lies in getting readable informations from systems represented by graphs. See, for example, Kaufmann and Wagner [4] and Di Battista [1] for a throughly survey in graph drawing. The edge-crossing minimization criterion is one of the most common to obtain a readable drawing. The problem of minimizing the number of crossings is NP-complete.

In this work we consider hierarchical graphs, this is not a limitation since there exists several methodologies to convert any directed acyclic graph (DAG) into a layered graph. The Hierarchical Directed Acyclical Graph (HDAG) representation is done by setting all the vertices in layers, and all the edges pointing in the same direction. The most well-known method to obtain a good representation of a graph is the Sugiyama's procedure [9], which has become a standard in the field.

We focus on minimizing the number of edge crossings in incremental graph drawings. The goal is to preserve the mental map of the user over successive drawings. In this way, we consider a new model adding constraints on both the relative ([7]) and the absolute position of the original vertices ([1] in the context of orthogonal graphs).

In this work, we propose a new mathematical programing formulation and a GRASP (Greedy Randomized Adaptive Search Procedure) with PR (Path Relinking) method. We adapt the well-known `Local Solver` black-box optimizer to solve this problem. Finally, we compare our heuristic with `CPLEX` and `Local Solver`, we have verified that `GRASP+PR` is able to obtain a fraction of the optimal solutions.

This work has been partially supported by the Spanish Ministerio de Economía y Competitividad with grant ref. TIN2015-65460-C02.

II. Mathematical Programing Model

A hierarchical graph $H = (G, p, L)$ is defined as a graph $G = (V, E)$ where V and E represent the sets of vertices and edges, respectively, p is the number of layers, and $L : V \rightarrow \{1, 2, \ldots, p\}$ is a function that indicates the layer where a vertex $v \in V$ resides. Let V^t be the set of vertices in layer t, and let E^t be the set of edges from V^t to V^{t+1}, with $n_t = |V^t|$.

The problem of minimizing edge-crossing is well-known in graph drawing [1] and, in the context of hierarchical graphs, may be formulated as the problem of finding the optimal ordering in each layer. A drawing D of a hierarchical graph H is the pair (H, Π) where Π is the set $\{\pi^1, \ldots, \pi^p\}$, with π^t establishing the ordering of the vertices in the layer t. We define as $C(D)$ the total number of edge crossings in drawing D.

From an original hierarchical graph $H = (G, p, L)$ and its drawing $D = (H, \Pi_0)$, we can consider the addition of some vertices $\hat{V}$ and edges $\hat{E}$ obtaining an incremental graph $IH = (IV, IE, p, L)$, where $IV = V \cup \hat{V}$ and $IE = E \cup \hat{E}$, p is the number of layers and m_t is the number of vertices in the incremental graph in layer t.

The goal in the Incremental Graph Drawing Problem (IGDP) is to minimize the number of edge crossings of an incremental drawing $ID = (IH, \Pi)$, while conserving the same relative position between the original vertices as in the original drawing D. The mathematical programing formulation of IGDP is based on a linear integer formulation for the Multi-Layer Drawing Problem proposed by Jünger et al. in [3], with a new set of constraints that preserves the relative position of the original vertices. As the authors in [3], we define the binary variables x_{ik}^t, where x_{ik}^t take the value 1 if vertex i precedes vertex k and 0 otherwise, where i, k are vertices that reside in the same layer t, $t = 1, \ldots, p$. Let $\pi_0^t(i)$ be the original position of the vertex i. The new constraints are:

$$x_{ij}^t + x_{ji}^t = 1, \quad \forall\, i, j \in V^t \ : 1 \leq i < j \leq m_t \qquad (1)$$

$$x_{ij}^t = 1, \quad \forall\, i, j \in V^t \ : \ \pi_0^t(i) < \pi_0^t(j). \qquad (2)$$

Finally, we add the requirement that the positions of the original vertices has to be set close enough to their positions in the original drawing. The maximum distance slack between the original position of a vertex and the new one, is represented by K. The mathematical model for the Constrained Incremental Graph Drawing Problem (C-IGDP) consists in the IGDP model adding the following set of constraints:

$$\max\{1, \pi_0^t(i) - K\} \leq \pi^t(i) \leq \min\{\pi_0^t(i) + K, m_t\}, \ \forall\ i \in V, \qquad (3)$$

where $\pi_0^t(i)$ and $\pi^t(i)$ represent the position of the vertex i, original and the new one where i can be assigned, respectively.

III. Greedy Randomized Adaptive Search Procedure

In this work, we propose a GRASP metaheuristic. The method iteratively performs two phases, constructive and improvement. The first phase employs a greedy function $g(v)$ that measures the minimum number of edge crossings $g(v, q)$ when vertex v is inserted in position q in its layer, for all positions q.

Initially, the construction phase starts with the original drawing, and iteratively, a vertex v is selected from the Restricted Candidate List (RCL) and it is placed in the position in which the number of crossings is minimum. The RCL consists of all unselected vertices (CL) with the number of edge crossings lower than or equal to a threshold τ:

$$\tau = \min_{v \in CL} g(v) + \alpha \left(\max_{v \in CL} g(v) - \min_{v \in CL} g(v) \right). \qquad (4)$$

That is $RCL = \{v \in CL : g(v) \leq \tau\}$. The performance of this method depends on the parameter α which balances greediness and randomness. Note that during the construction phase all the original vertices should maintain the restriction $\max\{1, \pi_0(v) - K\}$ and $\min\{\pi_0(v) + K, m_{L(v)}\}$. The pseudocode of construction phase is described in Fig. 1.

Once a solution ID is obtained, we apply our improvement method which consists in two neighborhoods $N_0(ID)$ and $N_1(ID)$. First, the method explores $N_0(ID)$, which consists in swapping the positions of two incremental vertices. For a incremental vertex v, from the first until the last incremental vertex in a layer, our procedure searches the best swap move with all the other incremental vertices, and performs it if it reduces the number of edge crossings. The algorithm scans all the swap moves from layer 1 to p until no further improvement is possible. Since only incremental vertices are involved, all the moves are feasible. Henceforth, we call this local search phase as *Swap*.

```
1  ID ← D;
2  CL ← IV \ V;
3  forall v ∈ CL do
4  |  compute g(v) and τ;
5  forall v ∈ CL do
6  |  if g(v) ≤ τ then
7  |  |  RCL ← RCL ∪ {v};
8  while ID is not complete do
9  |  v* ←select_node_randomly(RCL);
10 |  ID ← add_node_to_solution(v*);
11 |  recompute g and τ;
12 |  rebuild RCL;
13 return ID
```

Fig. 1. Constructive phase for the C-IGDP.

Then, the local search method resorts to a second phase, called *Insertion*, based on neighborhood $N_1(ID)$. In a given layer, the method scans all incremental vertices (starting from the first one) and explores all its feasible insertions in previous positions (a move is feasible if the position of the original vertices is within the limits of (3)). As in the case of $N_0(ID)$, the local search performs sweeps from layer 1 to p until no improvement is possible.

Our improvement procedure ends when *Swap* and *Insertion* phases are performed and a local optimum is found.

IV. Path Relinking

In this work, we propose a hybridization of GRASP with forward Path Relinking (PR). PR was originally proposed in the context of Tabu Search [2] to integrate search diversification and intensification. Later, Laguna and Martí in [6] adapted this technique as an intensification phase for GRASP. The method generates intermediate solutions in the path to connect high-quality solutions, which could be better or more diverse than the solutions being connected.

Let ID^i and ID^g be the initial and guiding incremental drawings, respectively. A path relinking move consists of replacing an entire layer from the initial solution ID^i with a layer from ID^g. Fig. 2 illustrates the path between two solutions ID^i and ID^g with 3 layers and 6 vertices. The three first intermediate solutions ID^1, ID^2 and ID^3 are generated from ID^i, exchanging one of its layer with the layers of ID^g. The process continues from the best solution found, in this case solution ID^1 with $C(ID^1) = 2$, in case of ties the algorithm choose one of the solutions at random. The algorithm finds two solutions, ID^4 and ID^5, that are better

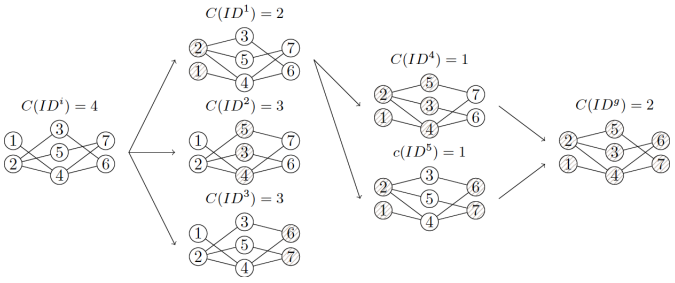

Fig. 2. Illustration of PR.

than ID^i and ID^g, both with number of edge crossings equal to 1.

The PR explores the path between all pairs of solutions in a set, called *Elite Set* (ES), which is constructed with $m = |ES|$ solutions generated with our procedure, and has the property of being as good as diverse as possible. Initially, the ES contains the m first solutions obtained with our GRASP. In each iteration, this set is updated with the generated solution ID^* if ID^* is better than the best solution in ES or sufficiently different from the other elite solutions. Diversity is measured as the number of positions that are not occupied by the same vertices divided by the number of vertices in the graph (with a distance larger than a parameter γ), while the quality is evaluated by the objective function. The algorithm ends when no new solutions are admitted to ES.

V. Experimental results

This section describes the experimental analysis to study the effectiveness and efficiency of the procedures presented above. In particular, we consider the following methods: GRASP (construction phase + local search), and GRASP+PR where we couple GRASP with Path Relinking. We also compare our heuristics with the solution obtained by running the integer linear programing proposed above with LocalSolver, and CPLEX.

The algorithms are implemented in C++, and the experiments have performed in an Intel Corei7-4020HQ CPU @2.60Ghz x 8. The set of instances consists of 240 files with 2, 6, 13 and 20 as the number of layers and the graph density varying in the range $\{0.065, 0.175, 0.3\}$. This set is generated in line with previous graph drawing papers [8] and [5], and it is available on-line in the web-page http://www.optsicom.es.

To avoid over-training, we consider 24 representative instances from the set of 240 to fine tune the algorithmic parameters. Note that, an important element is the value of the slack, K, between the original position of the vertices and the feasible ones. In this work, we consider low values of this parameter, $K = 1, 2$ and 3. Then, for each instance in the testing set, we have at most three different instances, one for each value of K (if the number of incremental vertices in one layer is less than K, then this value cannot be considered). To sum up, we have 609 instances in our *testing set*, and 62 instances in the *training set*.

These preliminary experimentation is devoted to fine tune the parameters in our GRASP+PR heuristic. First, the value of the parameter α in the construction phase which controls the balance between the diversity and the quality of the solution. We test three different values of α: 0.25, 0.5, and 0.75, and we also include a variant, labelled as $random$ that, for each construction, the value of α is selected by random in the range $[0, 1]$. The best performance is achieved with this $random$ variant. The last parameters to be set are the elite set size and the distance parameter in the PR procedure. We set these values to $|ES| = 3$ and $\gamma = 0.2$. The selection of the values of the parameters in these preliminary experiments is a trade-off between quality and computing time, for this reason, we do not reproduce the tables.

Table I shows the comparison between GRASP and GRASP+PR with CPLEX and LocalSolver over the entire set of 609 instances, classified by size. We execute our heuristics for 100 iterations, LocalSolver is run with a time limit of 20 seconds on the instances with 2 layers, and 60, 150 and 300 seconds on those with 6, 13 and 20 layers, respectively. Finally, we configure CPLEX to run for a maximum of 1800 seconds. This table shows for each procedure, the average number of crossings ($\bar{C}$), the average percent deviation from the best solution found ($\%\ dev$), the average percentage deviation between the heuristic solution value and the CPLEX best solution value ($\%\ gap$), the number of best solutions ($Bests$), the number of optimum solutions (Opt), and the CPU-time in seconds required to execute the method ($Time$).

TABLE I
Comparison on entire benchmark set according to instance size

Procedures	$\overline{C}$	$\%\ gap$	$\%\ dev$	$Bests$	Opt	$Time$
2 Layers ($32 \leq n \leq 96$), 171 instances						
CPLEX	2408.50	-	0.00	171	171	0.77
GRASP	2409.19	0.14	0.14	161	161	1.11
GRASP+PR	2408.92	0.14	0.14	164	164	1.18
LocalSolver	2785.47	17.01	17.01	12	12	20.11
6 Layers ($48 \leq n \leq 288$), 159 instances						
CPLEX	9995.70	-	0.01	157	157	56.24
GRASP	9997.43	0.13	0.14	81	81	5.53
GRASP+PR	9994.32	0.08	0.08	100	98	5.39
LocalSolver	11024.89	16.59	16.6	0	0	62.43
13 Layers ($104 \leq n \leq 611$), 141 instances						
CPLEX	23469.05	-	0.21	131	128	273.77
GRASP	23319.41	0.13	0.33	29	27	15.22
GRASP+PR	23305.22	0.02	0.22	54	44	15.34
LocalSolver	25530.24	15.95	16.16	0	0	162.06
20 Layers ($120 \leq n \leq 960$), 138 instances						
CPLEX	37918.20	-	0.38	118	116	383.73
GRASP	37522.42	0.03	0.39	21	20	25.77
GRASP+PR	37495.44	-0.12	0.24	49	29	28.39
LocalSolver	40954.07	14.30	14.68	0	0	328.77

Table I shows that, as expected, GRASP+PR obtains better results than GRASP. It is worth mentioning that CPLEX is able to obtain the optimal solution in 572 out of the 609 instances, and, with similar CPU time as our heuristics, it is able to obtain all the exact solutions for the small instances. However, in large instances, our heuristics are able to obtain similar quality

results as CPLEX with lower CPU time. Note that GRASP + PR is able to obtain a $\%\ gap$ of -0.12 which means that on average beats CPLEX. LocalSolver obtains the largest deviations in both $\%\ dev$ and $\%\ gap$, even if it is executed for longer CPU times than the competing heuristics.

VI. Conclusions

In this work, we propose a new mathematical model to tackle the Multilayer Incremental Graph Drawing Problem. We consider new constraints (by means of a paramenter K) to preserve key characteristics when updating an existing drawing. We develop a GRASP algorithm combined with a Path Relinking to obtain high quality solutions in the long term. We compare our heuristic with general solvers as CPLEX, which is able to obtain optimal solutions in 572 out of the 609 instances, and LocalSolver. GRASP+PR is competitive with them, obtaining similar quality solutions as CPLEX in smaller times on large instances, and outperforming LocalSolver.

References

[1] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1998.

[2] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[3] M. Jünger, E. K. Lee, P. Mutzel, and T. Odenthal. A polyhedral approach to the multi-layer crossing minimization problem. In *International Symposium on Graph Drawing*, pages 13–24. Springer, 1997.

[4] M. Kaufmann and D. Wagner, editors. *Drawing Graphs: Methods and Models*. Springer-Verlag, London, UK, UK, 2001.

[5] M. Laguna, R. Martí and V. Valls. Arc crossing minimization in hierarchical digraphs with tabu search. *Computers and Operations Research*, 24:1175–1186, 1997.

[6] M. Laguna and R. Martí. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

[7] R. Martí, A. Martínez-Gavara, J. Snchez-Oro, and A. Duarte. Tabu search for the dynamic bipartite drawing problem. *Computers & Operations Research*, 91:1–12, 2018.

[8] J. Sánchez-Oro, A. Martínez-Gavara, M. Laguna, A. Duarte, and A. Martí. Variable neighborhood scatter search for the incremental graph drawing problem. *Computational Optimization and Applications*, 68:775–797, 2017.

[9] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man, Cybern.*, 11:109–125, 1981.