Minimizing crossings in a hierarchical digraphs with a hybridized genetic algorithm

Pascale Kuntz, Bruno Pinaud, Rémi Lehn

Minimizing Crossings in Hierarchical Digraphs with a Hybridized Genetic Algorithm

Pascale Kuntz (pascale.kuntz@univ-nantes.fr)[1,2], Bruno Pinaud (bruno.pinaud@univ-nantes.fr)[1,2,3] and Rémi Lehn (remi.lehn@univ-nantes.fr)[2]
1-Ecole Polytechnique de l'université de Nantes - 44306 Nantes - France
2-Laboratoire d'informatique de Nantes Atlantique (LINA) - 44300 Nantes - France
3-Knowesia SAS - 44470 Carquefou - France

Abstract. Producing clear and intelligible layouts of hierarchical digraphs knows a renewed interest in information visualization. Recent experimental results show that metaheuristics are well-adapted methods for this problem. In this paper, we develop a new Hybridized Genetic Algorithm for arc crossing minimization. It follows the basic scheme of a GA with two major differences: problem-based crossovers adapted from ordering GAs are combined with a local search strategy based on averaging heuristics. Computational testing was performed on a set of 180 random hierarchical digraphs of standard sizes with various structures. Results show that the Hybridized Genetic Algorithm significantly outperforms Tabu Search -which is one of the best known methods for this problem- and also a multi-start descent except for highly connected graphs.

Keywords: Graph Drawing, Hierarchical Digraph, Genetic Algorithm, Metaheuristics

Introduction

The currently renewed interest in graph drawing is associated with increasing needs in information visualization (Herman et al., 2000). Beside areas of application traditionally cited (*e.g.* project management, circuit schematics, data structures, taxonomy), new fields have recently emerged including web cartography (Andrews, 1995; Kraak and Brown, 2000) and visualization supports in data mining (Buntine, 1996). A major advantage of graphs is that they can be used at the same time as abstract models for relations among data and as efficient visualization tools making access to complex structures easier without getting bogged down in theoretical concepts.

When producing clear and intelligible layouts, three key points are always considered (Di-Battista et al., 1999): the physical constraints inherent to the medium (standard size sheets, computer screen), the drawing conventions and the aesthetics. Drawing conventions essentially depend on the rules of the application domain. In this paper, we focus on hierarchical digraphs, also called layered digraphs where vertices are arranged on vertical or horizontal layers, and arcs linking vertex pairs are represented by oriented line segments which flow in the same direction. From the seminal works of Warfield, 1977 and Sugiyama et al., 1981, this representation has been chosen for

highlighting directed pairwise relations in numerous systems. Optimizing aesthetics aims at facilitating readability and memorization of the information embodied in the graph. Recent experiments confirming previous results from Carpano, 1980 indicate that edge crossings is by far the most important aesthetic criterion (Purchase, 2000).

Minimizing crossings in a hierarchical layout could seem intuitively easier than the general problem of minimizing crossings on a plane since the number of crossings is determined by the vertex ordering instead of the vertex geometric coordinates. Yet, it remains *NP*-complete even if there are only two layers (Garey and Johnson, 1983). Numerous deterministic heuristics follow the layer-by-layer sweep scheme: vertices of each layer are reordered to reduce crossings while holding the vertex orderings on the other layers. Various strategies have been proposed for reordering (see Laguna et al., 1997 for a chronological review). The most commonly used are the sorting methods and the averaging heuristics which include the barycenter heuristics (Sugiyama et al., 1981), the median heuristics (Eades and Wormald, 1994) and their variants (Gansner et al., 1988; Rowe et al., 1987). Sorting methods exchange vertices using crossing numbers in a way similar to classical sorts (Warfield, 1977). Averaging heuristics are based on the idea that edge crossings tend to be minimized when connected vertices are placed facing each other. Consequently, vertices are arranged according to their neighbor average positions, *e.g.* the arithmetic mean or the median.

Recently, the metaheuristics Tabu Search (Laguna et al., 1997) and GRASP (Greedy Randomized Adaptive Search Procedure) (Marti, 2001) have been developed for this problem. Numerical comparisons reported in Marti, 2001 show that Tabu Search gives better drawings than GRASP with a higher computational cost and that both search processes outperform the deterministic layer-by-layer sweep approaches. Moreover, a statistical analysis of the fitness landscapes associated with different local transformations has confirmed the presence of numerous local optima and advocates a highly stochastic approach able to move away from neighbors of a local optimum "late" in the search process (Kuntz et al., 2004). In addition to the metaheuristics mentioned above, the Genetic Algorithms satisfy these requirements (Goldberg, 1989). Promising experiments with Genetic Algorithms for different graph drawing problems (Groves et al., 1990; Mäkinen and Sieranta, 1994; Ochoa-Rodrìguez and Rosete-Suàrez, 1995; Utech et al., 1998; Tettamanzi, 1998) have encouraged us to explore this approach for hierarchical digraphs.

This paper describes a new hybridized Genetic Algorithm for arc crossing minimization in proper hierarchical digraphs. This algorithm combines problem-based crossover operators adapted from ordering Genetic Algorithms with a local search strategy based on averaging heuristics. We compare two local search strategies and precisely study their respective influence on the quality of the optimization process. We show that the local search that is

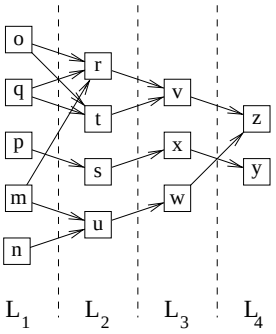

Figure 1. A hierarchical digraph with 4 layers.

then selected markedly improves the results obtained by a generic Genetic Algorithm. We compare our approach with both multi-start descent and Tabu Search which, as far as we know, is currently the best published metaheuristics for the problem considered. Through experiments on random graphs of different densities and sizes and compatible with drawings on standard size supports, we show that our approach significantly outperforms Tabu Search in most cases. This is also valid for multi-start descent for graphs of low density which are the most sensitive to additional crossings in terms of interpretation.

The rest of the paper is organized as follows: section 1 introduces a formal definition of the problem, section 2 describes the genetic operators and the hybridization, section 3 presents the experimental results, and a new application prospect of our approach is discussed in section 4.

1. Problem Description and Notations

Let $G = (V,A)$ be an acyclic digraph in which V is the vertex set and A is the arc set. The associated hierarchy $H_G = (G,L)$ is defined by a partition L of V into h layers $L_1, L_2, ..., L_h$ so that if $(u,v) \in A$, where $u \in L_i$ and $v \in L_j$, then $i < j$. The preliminary stage of layer assignment is here supposed to be done (see Di-Battista et al., 1999 for a description).

The span of an arc $(u,v) \in A$ with $u \in L_i$ and $v \in L_j$ is $j - i$. Without any effect, we can restrict ourselves to proper hierarchies with arc span exactly one. In practice, it is easy to come down to this hypothesis by replacing each arc whose span λ is greater than one by a path of $\lambda - 1$ dummy vertices on consecutive layers.

We use the straight-line convention where each arc is drawn as a straight line segment between layers represented by equally spaced vertical lines (Figure 1). The vertex ordering on each layer L_k is defined by
$\pi_k : L_k \rightarrow \{1, 2, ..., |L_k|\}$, where $\pi_k(u) = i$ means that the vertex $u \in L_k$ is on

the i^{th} position on L_k, and $\sigma_k(i) = \pi_k^{-1}(i)$ indicates the vertex on each position i. A drawing of H_G is a set of orderings $\Pi = \{\pi_1, \pi_2, ..., \pi_h\}$ on each layer of L. The crossing number associated with Π is denoted by $c(\Pi)$.

The problem of minimizing crossings consists in finding an optimal ordering set $\widehat{\Pi}$ so that there is no ordering set with fewer crossings.

2. The Hybridized Genetic Algorithm

2.1. Solution coding and general scheme

As the search space is composed of ordering sets, all operators are functions of an ordinal representation. In the genotype, the layers are considered one after the other from L_1 to L_h, and for each layer L_k, we code the vertex present at each position from 1 to $|L_k|$. By construction, a vertex cannot appear in more than one layer. The genotype associated with Π is defined by

$$g(\Pi) = (\sigma_1(1), ..., \sigma_1(|L_1|), \sigma_2(1), ..., \sigma_2(|L_2|), ..., \sigma_h(1), ..., \sigma_h(|L_h|))$$

The fitness function $f(g(\Pi)) = 2^{-c(\Pi)}$ is a usual function of the crossing number $c(\Pi)$. Without reference works on the influence of the fitness characteristics on the Genetic Algorithm convergence for drawing problems, this choice is rather arbitrary. However, its judiciousness seems to be confirmed when comparing it with the other function sometimes considered

$$f'(g(\Pi)) = 1 - \frac{c(\Pi) - c_{min}(\Pi)}{c_{max}(\Pi) - c_{min}(\Pi)}$$

where $c_{max}(\Pi)$ and $c_{min}(\Pi)$ are respectively the maximal and minimal crossing numbers obtained on the population. On the 180-graph set described in Section 3 with 100 runs per graph, 86.36% of the best solutions for HGA1 are obtained by f and this difference remains high whatever the graph density (89.1% for $d(H_G) = 0.3$, 85.4% for $d(H_G) = 0.5$ and 84.6% for $d(H_G) = 0.7$).

The algorithm follows the basic scheme of a Genetic Algorithm with two major differences: crossover brings two problem-based operators together and solutions built by the genetic operators are transformed by a local search (Algorithm 1). Let us note that we restrict ourselves to operators which lead to valid solutions only.

The selection is determined by a classical roulette wheel based on the fitness function $f(g(\Pi))$. The mutation is concerned by each layer with a probability p_{mut}; when it is applied on a layer L_k, two randomly chosen vertices of L_k are swapped. A generation is here a complete execution of the "while" statement in Algorithm 1; it leads to a renewed population since all parents are replaced by children.

Algorithm 1 Pseudo-code of the hybridized Genetic Algorithm (HGA)

Generate a random *Population* of N genotypes;
While ($\neg$termination criterion) {
 Initialize an empty population *Current_population;*
 for $i = 0, ..., \lceil N/2 - 1 \rceil$ {
  Select two genotypes $g(\Pi)$ and $g(\Pi')$ in *Population*;
  Apply Intra-Layer-Crossover on $g(\Pi)$ and $g(\Pi')$ to create $g(\Pi_i)$ and $g(\Pi'_i)$;
  Apply Inter-Layer-Crossover on $g(\Pi_i)$ and $g(\Pi'_i)$ to create $g(\Pi_c)$ and $g(\Pi'_c)$;
  Apply mutation on $g(\Pi_c)$ and on $g(\Pi'_c)$;
  Apply Local Search on $g(\Pi_c)$ and on $g(\Pi'_c)$;
  Add $g(\Pi_c)$ and $g(\Pi'_c)$ to *Current_population*;
  }
 Replace *Population* by *Current_population*;
 }

2.2. PROBLEM-BASED CROSSOVER OPERATORS

For the hierarchical drawing problem, the condition of improvement by recombination of building blocks is satisfied: the combination of two well-adapted drawings of sub-hierarchies can produce an even better adapted drawing and does not systematically lead to a damaged solution (figure 2). The combinations are applied between layers (inter-layer crossover) or inside each layer (intra-layer crossover).

Inter-layer crossover. The inter-layer crossover operator is a unique-point crossover between layers. Let i be a random integer in $\{1, 2, ..., h\}$. The resulting inter-layer crossover between the parents $g(\Pi) = (\sigma_1(1), ..., \sigma_h(|L_h|))$ and $g(\Pi') = (\sigma'_1(1), ..., \sigma'_h(|L_h|))$ gives two new children

$$(\sigma_1(1), ..., \sigma_{i-1}(|L_{i-1}|), \sigma'_i(1), ..., \sigma'_h(|L_h|))$$

and

$$(\sigma'_1(1), ..., \sigma'_{i-1}(|L_{i-1}|), \sigma_i(1), ..., \sigma_h(|L_h|))$$

Intra-layer crossover. Combining blocks inside a layer meets a well-known difficulty for ordinal codings which is to define a crossover which guarantees a feasible solution (Kargupta et al., 1992; Whitley and Yoo, 1995). The intra-layer crossover is a generalization of the Order Crossover 1 (Whitley and Yoo, 1995) for multi-permutations. A pivot p whose random position is normalized by the layer cardinality is defined for each layer. Above the pivot, vertex positions of the first genotype are kept and below, positions of

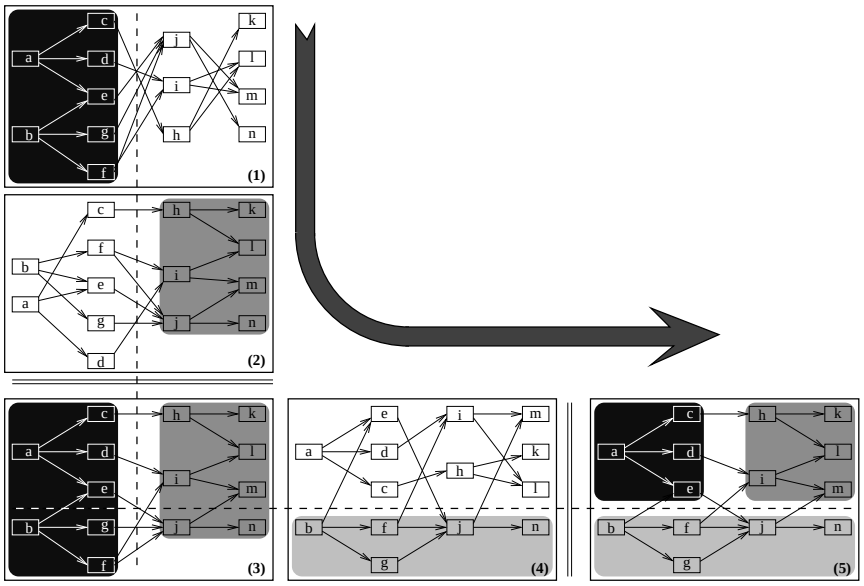

Figure 2. Improvement of an initial layout with successive sub-hierarchy combinations. The inter-layer combination of (1) and (2) produces (3) and the intra-layer combination of (3) and (4) produces (5) which is a global optimum.

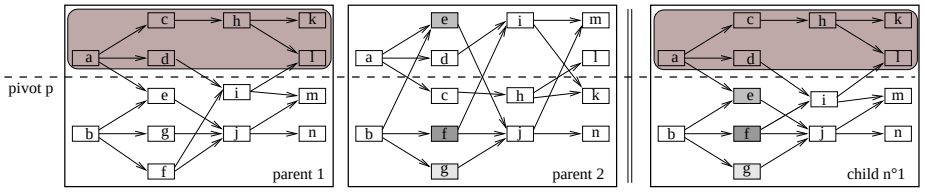

Figure 3. Example of an application of the intra-layer crossover for the first child only. The part of the graph above the pivot in the first parent $g(\Pi)$ is kept as it is. The child is then completed by the missing vertices in their order of appearance in the second parent $g(\Pi')$.

missing vertices are completed by those of the second genotype according to the vertex ordering. And vice-versa for the second child (see figure 3).

The pivot position p in a layer L_k is chosen uniformly at random from $\{1,2,...,|L_k|\}$. The resulting components for L_k of the intra-layer crossover between $g(\Pi)$ and $g(\Pi')$ are

$$\left(\sigma_k(1),...,\sigma_k(p),\rho'_k(1),...,\rho'_k(|L_k|-p)\right)$$

and

$$\left(\sigma'_k(1),...,\sigma'_k(p),\rho_k(1),...,\rho_k(|L_k|-p)\right)$$

where $\rho'_k(i)$ is the i^{th} vertex of $\left(\sigma'_k(1),\sigma'_k(2),...,\sigma'_k(|L_k|)\right)$ not present in $(\sigma_k(1),\sigma_k(2),...,\sigma_k(p))$ and similarly, $\rho_k(i)$ is the i^{th} vertex of $(\sigma_k(1),\sigma_k(2),...,\sigma_k(|L_k|))$ not in $\left(\sigma'_k(1),\sigma'_k(2),...,\sigma'_k(p)\right)$.

These two crossovers are successively applied with the probabilities p_{intra} and p_{inter}. Given the parents $g(\Pi)$ and $g(\Pi')$, they are first transformed by the intra-layer crossover into two intermediate genotypes $g(\Pi_i)$ and $g(\Pi'_i)$, which themselves produce two new children $g(\Pi_c)$ and $g(\Pi'_c)$ with the inter-layer crossover.

2.3. LOCAL SEARCH

Finding a local search strategy which globally leads to better results without too much deteriorating the computation time usually requires numerous trials and errors on the basis of a variety of test cases. In the case of a local improvement process the favored approach is the steepest descent when the neighborhood of the current solution can be easily defined. For graph layouts different metrics could be investigated to define a topology (Bridgeman and Tamassia, 2002). But, they often are time-consuming, and the selection of the most suitable one for the optimization process remains an open issue. An alternative approach is to resort to problem-based local heuristics. The averaging heuristics reported in the introduction are known for computing solutions in a short time. Although more complex approaches coming originally from circuit design have been recently proposed (Eschbach et al., 2002; Matuszewski et al., 1999), we here restrict ourselves to the original ones which are the most popular. Hence, the considered local search strategy combines the greedy-switch heuristics and adaptive versions of the median and the barycenter heuristics for each layer. We describe them for a given layer L_k.

The greedy-switch heuristics switches random consecutive pairs of vertices.

The median position $m(u)$ of a vertex u on L_k is here a function of the connected vertices in both layers L_{k-1} and L_k. Let us denote by $v_1,v_2,...,v_p$ the neighbors of u on L_{k-1}, by $w_1,w_2,...,w_q$ the neighbors on L_{k+1}, and by $N(u)$ the set of their normalized positions defined by

$$N(u) = \left(\frac{\pi_{k-1}(v_1)}{|L_{k-1}|},...,\frac{\pi_{k-1}(v_p)}{|L_{k-1}|},\frac{\pi_{k+1}(w_1)}{|L_{k+1}|},...,\frac{\pi_{k+1}(w_q)}{|L_{k+1}|} \right)$$

The set $N(u)$ is arranged in increasing order and the median is the number $m(u) \in N(u)$ so that half of the elements of $N(u)$ are less or equal than $m(u)$ and half are greater than or equal to $m(u)$. The new vertex ordering π^m_k on L_k results from the median arrangement: for any u, $v \in L_k$, $\pi^m_k(u) \geq \pi^m_k(v)$ if and only if $m(u) \geq m(v)$.

In some instances, we have experimentally observed that solutions are slightly improved when the barycenter heuristics is performed after the median heuristics. This may be explained by a combination of two effects. It is well-known that for some graphs one of the two heuristics is better suited (Di-Battista et al., 1999). Moreover, there is a side effect of the chosen median definition which does not explicitly take the parity of $N(u)$ into account.

The barycenter of $u \in L_k$ is defined by the arithmetic mean of the normalized positions of the connected vertices of u on L_{k-1} and L_{k+1}:

$$b(u) = \frac{\sum_{v_i} \frac{\pi_{k-1}(v_i)}{|L_{k-1}|} + \sum_{w_i} \frac{\pi_{k+1}(w_i)}{|L_{k+1}|}}{|L_{k-1}| + |L_{k+1}|}$$

As above, the new vertex ordering π_k^b is deduced from the barycenter arrangement: for any $u, v \in L_k$, $\pi_k^b(u) \geq \pi_k^b(v)$ if and only if $b(u) \geq b(v)$.

Considering both adjacent layers L_{k-1} and L_{k+1} together has proved less performing in the Sugiyama's heuristics (Sugiyama et al., 1981) than considering each adjacent layer separately in different order. The reason put forward was that the definition including both layers together depends on vertices (of L_{k+1}) whose orders have not yet been improved. However, the situation here differs since the averaging heuristics are not used alone but in combination with a GA. Moreover, the normalization introduced in our definitions, which is unusual in the literature, is motivated by the fact that we consider an ordinal coding for the genotypes instead of the real coordinates of the vertices used in other approaches.

On the basis on these local transformations, we have investigated two local search processes: LS1 which follows a strategy of local exploration of the search space, and LS2 which corresponds to a local optimization. In both processes, the greedy-switch, median and barycenter transformations are successively computed on the layer set with their respective probabilities p_{greed}, p_{med} and p_{bar}. For LS1 there is no intermediate evaluation of the obtained solutions; in the associated hybridized GA the evaluation is put off till the selection stage. For LS2 the drawing resulting from the application of a transformation on the layer set is only preserved when crossings decrease. In the following, we denote by HGA1 (resp. HGA2) the hybridized GA associated with LS1 (resp. LS2) and by HGA the general hybridation strategy when there is no ambiguity (see Algorithm 1).

3. Computational Experiments

Computational testing was performed on a set S_{test} of 180 random graphs of various structures including those of real-life applications. Our random

hierarchical digraph generator works like the one previously developed for tests with metaheuristics (Laguna et al., 1997; Marti, 2001). Three parameters can vary: the number of layers, the number of vertices per layer and the graph density. A commonly used density measure is the ratio of the arc number m to the arc number of a complete graph of the same size, but we prefer a more suitable definition for hierarchical digraphs: as the maximal hierarchical digraph has

$$m_{max} = \sum_{k=2}^{h} |L_{k-1}| \times |L_k|$$

arcs, the hierarchical density is $d(H_G) = m/m_{max}$. The generator was used to create 20 instances for each combination of 4, 8 and 12 layers and 0.3, 0.5 and 0.7 hierarchical densities. The vertex number per layer is randomly chosen between 5 and 15. These sizes are well-representative for real-life applications where vertices are often represented by labelled boxes and can rarely exceed 70 or 80 on a standard size sheet. For large graphs, not considered here, alternative approaches (e.g. pre-partitioning, "overview and zoom", *etc.*) are implemented in practice (Mutzel and Jünger, 2003).

To compare the different approaches, we use the "best solution frequency": for each run of k algorithms A_1, ..., A_k on the same graph we count the number of runs for which each A_i has been the only one to reach the best solution. We additionnally consider identity cases where different algorithms reach the same solution; this information, often neglected in the literature, has here been proved relevant.

3.1. ALGORITHM PARAMETERS

Preliminary experiments reported in Appendix A, with 100 runs for each graph of S_{test} (total of 18000 runs), have shown that the best parameter values are very close for HGA1 and HGA2. In computational experiments with HGA1 (resp. HGA2), we have chosen the following operator probabilities: $p_{mut} = 0.02$ for mutation, $p_{intra} = p_{inter} = 0.2$ for crossovers and $p_{greed} = 0.05$, $p_{bar} = p_{med} = 0.2$ (resp. $p_{mut} = 0.02$ for mutation, $p_{intra} = p_{inter} = 0.25$ for crossovers and $p_{greed} = 0.1$, $p_{bar} = p_{med} = 0.3$). The results in Appendix A confirm that small variations around these values do not significantly modify either the result quality nor the computation time.

The population size has been fixed to 100. The algorithms stop when no improvement has occurred after 100 generations. Table I shows that a larger population and a greater generation number give slightly better results. But the choice of the population size is a compromise solution between quality and computational time which is a critical parameter for graph drawing.

HGA was coded in C and executed on an AMD Athlon at 2GHz with Linux 2.6.

Table I. Comparison of the best computed solution frequencies (in %) for different generation numbers (a) and different population sizes (b) for HGA1. 100 runs for each graph of S_{test} have been performed. Results include identity cases.

Number of generations	50	100	150	200	300
Best solution frequency	28.61	35.1	39.11	42.18	47.16
Computation time	1.45	2.57	3.64	4.66	6.72

a. Number of generations

Number of individuals	50	100	150	200	250	300
Best solution frequency	24.77	31.06	35.51	39.15	41.3	43.97
Computation time	1.33	2.57	3.79	5.03	6.19	7.29

b. Population size

Table II. Comparison of the best computed solution frequencies in % for HGA1 and HGA2.

	HGA1	HGA2	HGA1=HGA2
$d(H_G) = 0.3$	64.02	23.55	12.43
$d(H_G) = 0.5$	60.43	25.75	13.82
$d(H_G) = 0.7$	54.8	26.4	18.8

3.2. COMPARISON OF THE LOCAL SEARCH STRATEGIES

Table II shows that the hybridized HGA1 outperforms HGA2 whatever the graph characteristics. The performance difference is confirmed by a statistical test: the *p*-value for the Wilcoxon signed rank test for pair samples on the crossing numbers is very low (7.885×10^{-10} for a standard level of significance $\alpha = 0.05$). The computation time for the chosen stopping criterion (no improvement after 100 generations) is greater for HGA2 (3.8 seconds) than for HGA1 (2.6 seconds). This difference is due to the cost of the transformation evaluations in LS2. When picking out the first generation embodying the final solution, we observe that HGA2 converges faster on average than HGA1. In fact, for HGA2, the application of the local optimization reduces the positive effect of the exploration stage of the GA; it converges towards a solution of lower quality too quickly. Consequently, in the following, we retain HGA1 for all the comparisons.

3.3. INFLUENCE OF THE MAIN OPERATORS

We successively study the contributions to HGA1 of the local search and of the purely genetic part.

Adding a local search step in the classical Genetic Algorithm scheme has a great influence on three major characteristics of the optimization process: improving the convergence towards the best solution, saving the computation time and reducing the variability of the results inherent to the stochastic approaches. When comparing HGA1 with GA, *i.e.* HGA1 without local search, the best solution is obtained by HGA1 in 88.6% of the cases (see Table III-a for details), and the performance difference is confirmed by the Wilcoxon signed rank test (p-value $< 2.2 \times 10^{-16}$ for $\alpha = 0.05$). The average computation time saving $(t_{GA} - t_{HGA1})/t_{GA}$ is equal to 28.5% and increases with the density: 21.1% for $d(H_G) = 0.3$, 29.4% for $d(H_G) = 0.5$ and 35.2% for $d(H_G) = 0.7$. The benefit in stability of HGA1 is measured by $(\overline{\sigma}_{GA} - \overline{\sigma}_{HGA1})/\overline{\sigma}_{GA} = 53.2\%$, where $\overline{\sigma}$ is the average on the graph set of the crossing number standard deviations obtained for 100 runs.

To evaluate the importance of the part played by the genetic process, we compare HGA1 with a multi-start descent (MSD1) where the local search LS1 of HGA1 runs by itself several times from random initial points. More precisely, from a random drawing, LS1 is applied (with $p_{greed} = p_{mut} = p_{bar} = 1$) and, following a descent strategy, the solution is kept when crossings decrease. The algorithm stops when there is no more improvement.

For the comparison of HGA1 and MSD1 two criteria are considered: (MSD1-a) a random generation of a set of a given number of arrangements -here fixed to the usual yardstick value 100- and (MSD1-b) a stopping time equal to the convergence time of HGA1 towards the best computed solution. Table III-b shows that 100 local searches are far from being sufficient to outperform HGA1. With MSD1-b, the average number of arrangements is equal to 677.58. And, for this comparison, the best solutions are more frequently computed by HGA1 when $d(H_G) = 0.3$ and 0.5. But when $d(H_G) = 0.7$ the difference is no more significant. Moreover, in this last case, let us remark that 13.5% of the best solutions are simultaneously obtained by the two approaches.

When considering the exploration of the search space, it is interesting to compare the number of evaluations required by the two approaches. On average on the test set, MSD1 requires 5.27 "basic loops" per descent *i.e.* successive applications of the three local operators on the level set. Consequently, the mean total number of evaluations is equal to $5.27 \times 677 = 3567.79$. For HGA1 the mean total number of evaluations is equal to the population size by the generation number *i.e.* 9100. Hence, during the same computation time, HGA1 explores 2.55 times as many solutions as MSD1.

When the discovery of the global optimum cannot be guaranteed, the quality of the computed local optima is an important factor for applications. Generally speaking, when the best solution Π_M is found by a metaheuristics M different from HGA1, we compare the quality of the latter with the quality of the HGA1 solution Π_{HGA1}: the relative height $h_M(\Pi_{HGA1})$ of the solution Π_{HGA1} is defined by

$$h_M(\Pi_{HGA1}) = 1 - \frac{c(\Pi_{HGA1}) - c(\Pi_M)}{c(\Pi_M)}$$

If $h_M(\Pi_{HGA1})$ is very close to 1, then the quality of the drawing is often satisfactory for relevant applications, in particular for graphs with high densities where adding a few crossings is not always perceptible.

For graphs with density 0.7, the distribution of $h_{MSD1}(\Pi_{HGA1})$ is concentrated on the interval $[0.96, 1]$. These results allow us to conclude that even for a high density HGA1 remains efficient. However, completing a previous analysis on the fitness landscape structure associated with the local search operators (Kuntz et al., 2004), these results lead us to conjecture that for this graph class there are numerous local optima similar in value. Their distribution in the fitness landscape does not allow to fully take advantage of the efficiency of the Genetic Algorithm implicit learning. In this case, random searches, cheap in computing time, provide an exploration on a large scale.

3.4. Comparisons with Tabu Search

We compare HGA1 with the Tabu Search procedure (TS) of Laguna et al., 1997. Based on the same coding as HGA, the TS method links two stages together. First, a local optimum is computed layer-by-layer by vertex swapping. When the ordering of a layer can no more be enhanced, the layer is put in the tabu list as long as its adjacent layers are not modified. The process is repeated until all layers are in the tabu list. Then, for each local optimum, a local search is implemented by swapping two randomly chosen adjacent vertices $25 \times |V|$ times. A barycenter operator is applied for equality cases. The process terminates when the crossings number has not been enhanced after 50 consecutive runs of the complete procedure.

Table IV clearly points out that HGA1 is more efficient than TS. The performance difference between these two algorithms is confirmed by the Wilcoxon signed rank test. For the rare cases where TS outperforms HGA1, Table V shows that the HGA1-solution are not very far from those found by TS. On the other hand, when HGA1 computed the best solution, TS-solution are debased, especially for low density graphs.

Table III. Average in % of the best computed solution frequency between HGA1, a genetic approach without local search (GA) and a multi-start descent (MSD1) with two stopping criteria.

	HGA1	GA	HGA1=GA
$d(H_G) = 0.3$	88.58	8.35	3.13
$d(H_G) = 0.5$	81.9	14.22	3.88
$d(H_G) = 0.7$	72.28	18.02	9.7

a. Best computed solution frequency between HGA and GA.

	HGA1	MSD1-a	MSD1-b
$d(H_G) = 0.3$	48.6	5.5	30.42
$d(H_G) = 0.5$	47.8	5.43	31.27
$d(H_G) = 0.7$	37.67	5.83	36.02

	HGA1=MSD1-a	HGA1=MSD1-b	MSD1-a=MSD1-b	HGA=MSD1-a=MSD1-b
$d(H_G) = 0.3$	1.5	3.27	5.4	5.32
$d(H_G) = 0.5$	0.65	2.13	6.33	6.39
$d(H_G) = 0.7$	1.03	1.87	6.02	11.56

b. Best computed solution frequency between HGA1, MSD1-a and MSD2-b. The row HGA1=MSD1-a means that both heuristics find a solution with the same crossing number.

4. Conclusion

We have developed a hybridized Genetic Algorithm for straight-line crossing minimization in hierarchical drawings of digraphs. Computational comparisons on a set of random graphs of different densities show that this approach outperforms one of the best known metaheuristics, and that the solutions are either better or very close to those of a multi-start descent. For applicative issues focused on drawings on standard size supports, our results prove the great interest of the developed approach. Yet, they lead to an important open theoretical question concerning the link between the performances of the genetic algorithm and the structure of the fitness landscape. Our experiments seem to suggest that from a certain threshold of density there is a change in the distribution of the local optima which limits the benefits of the exploratory phase of the metaheuristics integrating learning capabilities. In the near future, we plan to study the correlation between statistical measures of the fitness landscape complexity and the influence of the genetic operators of the optimization process on large highly connected graphs.

On the other hand, beyond these results, we believe that an important extension of the genetic approach presented in this paper concerns the dynamic drawing problem, which has become an important stake in the development

Table IV. Direct comparison of HGA1 and TS for different densities $d(H_G)$. Distribution in percentage of the best computed solution frenquency.

$\|L\|$	4	8	12	Avg
HGA1	100	100	87.5	95.83
TS	0	0	8.33	2.78
HGA1=TS	0	0	4.17	1.39

a. Results for $d(H_G) = 0.3$

$\|L\|$	4	8	12	Avg
HGA1	91.3	88.89	94.44	91.54
TS	4.35	11.11	5.56	7.01
HGA1=TS	4.34	0	0	1.45

b. Results for $d(H_G) = 0.5$

$\|L\|$	4	8	12	Avg
HGA1	79.17	77.78	83.33	80.09
TS	8.33	22.22	16.67	15.74
HGA1=TS	12.5	0	0	4.17

c. Results for $d(H_G) = 0.7$

Table V. Average heights of the solution computed by HGA1 when TS computed the best solution ($h_{TS}(HGA1)$) and the average heights of the solution computed by TS when HGA1 computed the best solution ($h_{HGA1}(TS)$).

$d(H_G)$	0.3	0.5	0.7
$h_{TS}(HGA1)$	0.991	0.986	0.994
$h_{HGA1}(TS)$	0.848	0.955	0.977

of interactive graphic interfaces in information visualization. With most of the current algorithms, if a modification is performed on the graph, the algorithm runs again and produces a new drawing which may be thoroughly different from the previous one. But, as noticed by Papakostas et al., 1997, this is a waste of human resources to continually re-analyse the entire drawing and also of computational resources to re-compute the entire layout after each modification. The dynamic hierarchical drawing problem can be set as a

multi-objective problem: producing at each step t a drawing Π_t of the hierarchy H_t that satisfies the readability requirements, and so that Π_t remains as close as possible to the drawing Π_{t-1} of the previous hierarchy H_{t-1} to preserve the user's mental map. Different formalizations of this last constraint have been proposed (Eades et al., 1991; North, 1996). Results obtained in this paper, both on solution quality and computation time, as well as the recent advances in multi-criteria optimization with metaheuristics (C. A. Coello Coello et al., 2002) indicate that the Genetic Algorithms are especially good candidates for this problem.

Acknowledgements

The authors gratefully acknowledge the reviewers for their enlightening comments for the three revisions of this paper.

References

Andrews, K.: 1995, 'Visualizing cyberspace: Information visualization in the Harmony Internet browser'. In: *Proc. IEEE Symp. Inform. Visual.* pp. 97–105, IEEE Press.

Bridgeman, S. and R. Tamassia: 2002, 'A user study in similarity measures for graph drawing'. *J. of Graph Algorithms and Applications* **6**(3), 225–254.

Buntine, W.: 1996, 'Graphical models for knowledge discovery'. In: U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. S. Uthurasamy (eds.): *Advances in Knowledge Discovery and Data Mining*. MIT Press, pp. 59–83.

C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont: 2002, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Vol. 5 of *Genetic Algorithms and Evolutionary Computation*. Kluwer Academic Publishers. ISBN 0-3064-6762-3.

Carpano, M.: 1980, 'Automatic display of hierarchized graphs for computer aided decision analysis'. *IEEE Trans. Syst., Man, Cybern.* **10**(11), 705–715.

Di-Battista, G., P. Eades, R. Tamassia, and I. Tollis: 1999, *Graph drawing - Algorithms for the visualization of graphs*. Prentice Hall.

Eades, P., W. Lai, K. Misue, and K. Sugiyama: 1991, 'Preserving the mental map of a diagram'. In: *Proc. of Compugraphics*. pp. 24–33.

Eades, P. and N. Wormald: 1994, 'Edge crossings in drawings of bipartite graphs'. *Algorithmica* **11**, 379–403.

Eschbach, T., W. Günther, R. Drechsler, and B. Becker: 2002, 'Crossing reduction by windows optimization'. In: *Proc. of the* 10^{th} *Int. Symposium on Graph Drawing*. pp. 285–294, LNCS 2528, Springer Verlag.

Gansner, E., S. North, and K. Vo: 1988, 'A program that draws directed graphs'. *Software - Practice and Experience* **18**(11), 1047–1062.

Garey, M. and D. Johnson: 1983, 'Crossing number is *NP*-complete'. *J. Algebraic Discrete Methods* **4**(3), 312–316.

Goldberg, D.-E.: 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Groves, L.-J., Z. Michalewicz, P.-V. Elia, and C.-Z. Janikow: 1990, 'Genetic Algorithms for Drawing Directed Graphs'. In: *Proc. of the 5th Int. Symp. on Methodologies for Intelligent Systems*. pp. 268–276, Elsevier.

Herman, I., G. Melançon, and S. Marshall: 2000, 'Graph visualization and navigation in information visualization: a survey'. *IEEE Trans. Visual. Comput. Graphics* **6**(1), 24–43.

Kargupta, H., K. Deb, and D. Goldberg: 1992, 'Ordering genetic algorithms and deception'. In: *Proc. Parallel Problem Solving from Nature*, Vol. 2. pp. 47–56, Elsevier Sc.

Kraak, M. J. and A. Brown: 2000, *Web Cartography: Developments and Prospects*. Taylor&Francis. ISBN 0748408681.

Kuntz, P., B. Pinaud, and R. Lehn: 2004, 'Elements for the description of fitness landscapes associated with local operators for layered drawings of directed graphs'. In: M. Resende and J. de Sousa (eds.): *Metaheuristics: Computer Decision-Making*, Vol. 86 of *Applied optimization*. Kluwer Academic Publishers, pp. 405–420.

Laguna, M., R. Marti, and V. Valls: 1997, 'Arc crossing minimization in hierarchical digraphs with tabu search'. *Computers and Operation Research* **24**(12), 1175–1186.

Mäkinen, E. and M. Sieranta: 1994, 'Genetic Algorithms for Drawing Bipartite Graphs'. *Int. J. of Computer Mathematics* **53**(3-4), 157–166.

Marti, R.: 2001, 'Arc crossing minimization in graphs with GRASP'. *IIE Trans.* **33**(10), 913–919.

Matuszewski, C., R. Schönfeld, and P. Molitor: 1999, 'Using sifting for k-layer straightline crossing minimization'. In: *Proc. of the 7th Int. Symposium on Graph Drawing*. pp. 217–224, LNCS 1731, Springer Verlag.

Mutzel, P. and M. Jünger: 2003, *Graph drawing software*. Springer Verlag. ISBN: 3-540-00881-0.

North, S.: 1996, 'Incremental Layout in Dynadag'. In: *Proc. of Graph Drawing'95*, Vol. 1027 of *Lecture Notes in Computer Science*. pp. 409–418, Springer-Verlag.

Ochoa-Rodrìguez, A. and A. Rosete-Suàrez: 1995, 'Automatic Graph Drawing by Genetic Search'. In: *Proc. of the 11th Int. Conf. on CAD, CAM, Robotics and Manufactories of the Future*. pp. 982–987.

Papakostas, A., J. Six, and I. Tollis: 1997, 'Experimental and theoretical results in interactive orthogonal graph drawing'. In: *Proc. of Graph Drawing'96*, Vol. 1190 of *Lecture Notes in Computer Sciences*. pp. 371–386, Springer Verlag.

Purchase, H.: 2000, 'Effective information visualisation: a study of graph drawing aesthetics and algorithms'. *Interacting with computers* **13**(2).

Rowe, L., M. Davis, E. Messinger, C. Meyer, C. Spirakis, and A. Tuan: 1987, 'A browser for directed graphs'. *Software - Practice and Experience* **17**(1), 61–76.

Sugiyama, K., S. Tagawa, and M. Toda: 1981, 'Methods for visual understanding of hierarchical systems'. *IEEE Trans. Syst., Man, Cybern.* **11**(2), 109–125.

Tettamanzi, A. G.: 1998, 'Drawing graphs with evolutionary algorithms'. In: *Proc. of Conf. on Adaptive Computing in Design and Manufacture, ACDM*. pp. 325–338.

Utech, J., J. Branke, H. Schmeck, and P. Eades: 1998, 'An evolutionary algorithm for drawing directed graphs'. In: *Proc. of the Int. Conf. on Imaging Science, Systems and Technology*. pp. 154–160, CSREA Press.

Warfield, J.: 1977, 'Crossing theory and hierarchy mapping'. *IEEE Trans. Syst., Man, Cybern.* **7**(7), 505–523.

Whitley, D. and N. Yoo: 1995, 'Modeling simple genetic algorithms for permutation problems'. In: L. D. Whitley and M. D. Vose (eds.): *Fundations of Genetic Algorithms III*. pp. 163–184, Morgan Kaufmann.

Appendix

A. Parameter justification

Table VI. Distribution in % of the best solution frequencies for different parameter sets for HGA1. 100 runs for each graph of S_{test} have been performed. Results include identity cases where the best solutions are reached with different parameter sets. This explains that the sum is far greater than 100%.

p_{greed}	0.05	0.05	0.1	0.1	0.1
p_{bar}	0.1	0.2	0.1	0.2	0.3
p_{med}	0.1	0.2	0.1	0.2	0.3
Best solution frequency	33.5	41.17	28.4	35.7	40.42
Mean computation time (std dev)	2.46 (2.56)	2.58 (2.62)	2.54 (2.7)	2.63 (2.7)	2.78 (2.8)

a. Results for different probabilities of the local search operators

p_{intra}	0.2	0.25	0.3
p_{inter}	0.2	0.25	0.3
Best solution frequency	48.15	48.4	47.9
Mean computation time (std dev)	2.58 (1.78)	2.57 (1.76)	2.58 (1.73)

b. Results for different probabilities of the crossover operators

Table VII. Distribution in % of the best solution frequencies for different parameter sets for HGA2. 100 runs for each graph of S_{test} have been performed. Results include identity cases where the best solutions are reached with different parameter sets.

p_{greed}	0.05	0.05	0.1	0.1	0.1
p_{bar}	0.1	0.2	0.1	0.2	0.3
p_{med}	0.1	0.2	0.1	0.2	0.3
Best solution frequency	23.05	33.66	21.83	30.81	38.71
Mean computation time (std dev)	3.66 (3.17)	3.7 (3.13)	3.64 (3.14)	3.70 (3.12)	3.80 (3.16)

a. Results for different probabilities of the local optimization operators

p_{intra}	0.2	0.25	0.3
p_{inter}	0.2	0.25	0.3
Best solution frequency	41.76	42.87	41.47
Mean computation time (std dev)	3.70 (3.13)	3.69 (3.12)	3.69 (3.12)

b. Results for different probabilities of the crossover operators