A Hybrid Evolutionary Algorithm for the Edge Crossing Minimization Problem in Graph Drawing

Sergio Enríquez[1], Eunice Ponce de León[1], Elva Díaz[1] and Alejandro Padilla[1]

[1] Departamento de Ciencias de la Computación,
Centro de Ciencias Básicas,
Universidad Autónoma de Aguascalientes
{senriquez, eponce, ediazd, apadilla}@correo.uaa.mx

Abstract. This document explains the design and implementation of a hybrid evolutionary algorithm for the edge crossing minimization problem in graph drawing. This algorithm, called HUx, combines a global search algorithm (EDA — Estimation of Distribution Algorithm) with a local search Algorithm (HC — Hill Climbing) in order to establish a balance between the exploration and exploitation efforts. The HUx has shown to be more efficient and robust in search of the optimum solution in comparison to other meta-heuristics, such as HC, Univariate Marginal Distribution Algorithm and the Genetic Algorithm, which was also implemented in this comparison. Experiments were performed using planar and non-planar graphs. The quality and frequency of the optimum solutions registered by the hybrid HUx algorithm were higher than those registered by other algorithms.

Keywords: Graph Drawing, Crossing Minimization, Hill Climbing, Estimation of Distribution Algorithms, Genetic Algorithms.

1 Introduction

The problem in finding the minimum number of crossing edges in a graph entails finding, within all possible forms of drawing a graph, the one that includes the minimum number of crossings on its edges. The problem is difficult because as Garey and Johnson [7] showed is NP-hard. One of the main problems involved in the clear visualization of a graph is the problem of the cross minimization of the graph's edges [12]. This is a typical and very important problem that arises in graph drawing. Readability is reduced in graphs with a large number of crossings, especially in the case of dense graphs. The graph visualization problem is currently open to research; it can be applied to different areas, as in biology and chemistry, object oriented systems, data structures, real time systems, flowcharts, entity relation charts, semantic nets, project management, representation of knowledge charts, logical programming, design of VLSI circuits, virtual reality, cartography, and social networks, among others [8].

Received 30/04/10
Final version 19/05/10
Accepted 14/05/10

There are different Evolutionary Algorithms (EAs) which deal with the edge crossing minimization problem in graph drawing; however, most of these are based on genetic algorithms. The Estimation of Distribution Algorithms (EDAs) [10] has been poorly used in Graph Drawing (GD), for example, the only EDA algorithm known by the authors is UMDA in [14].

In this paper a hybrid algorithm composed with a global search algorithm (UMDA— Univariate Marginal Estimation Algorithm) and a local search algorithm (HC— Hill Climbing) is introduced. The HUx (Hill Climbing and UMDA x-crossing) has proven more efficient with regard to optimum solution search (in this case, the drawing which includes the least number of crossings on the graph's edges) than metaheuristics HC, UMDA and the Simple Genetic Algorithm (SGA) which was also implemented in this comparison. Measurement of results was carried out using different planar (with no crossings among its edges) and non-planar graphs (with at least one crossing among its edges), which were the measurement subjects to which the various already mentioned metaheuristics were applied. The quality and frequency of optimum solutions registered by the hybrid HUx algorithm were higher than those registered by the algorithms against which this measurement was carried out and that forms part of the state of the art within graph drawing. It must be mentioned that in each case studied, the hybrid HUx metaheuristic always finds the optimum for benchmarks and, additionally, the occurrence frequency of the optimum was 15% greater than the second best algorithm of all those studied. A proper graphical interface called Minx (Minimization x-crossing) was developed for graph visualization. This interface shows the automatic drawing of each graph obtained by the metaheuristics implemented in this paper.

2 Contents

The contents in this document are intended to:

(1) Define the optimization problem of the edge crossing minimization problem in graph drawing (section 3)

(2) Define hybrid base algorithms: UMDA algorithm and HC algorithms and their pseudo-code (section 4)

(3) Define the implemented hybrid HUx algorithm (section 5)

(4) Define how experiment designs were implemented (section 6)

(5) Compare results among the various implemented algorithms, and to compare results with the hybrid HUx algorithm results (section 7)

(6) Discuss and draw conclusions with regard to main research's contributions (section 8)

3 Optimization problem, solution representation and evaluation function

The graph edges crossing number optimization problem can be described as follows [5]:

Let $G = (V, E)$ be a graph, let V be a set of vertices and E a set of edges. MG is the adjacency matrix and P the Cartesian plane. The problem involves finding a pair $(x, y) \in P$ for each $v \in V$ such that the number of crossings between the edges of the graph is minimum. Each pair (x, y) represents a position of the vertex v in Cartesian Plane P. For any two vertices $v = (x, y)$ $v' = (x', y')$, x ? x' and y ? y'. In this paper, the graph's edges are considered straight lines.

Let N be the number of vertices of the graph. The solution representation S (i.e., a graph draw, or graph layout) is as follows:

$S=(x_1, y_1, x_2, y_2,\dots, x_i, y_i,\dots, x_N, y_N)$

Each pair x_i, y_i represents the position of the i-th vertex in Cartesian Plane P.

A graph draw S is evaluated by number of crossings of graph edges in this graph draw S. The number of crossings was obtained by solving an equation system for all pair of edges of the graph. The following cases are analyzed: crossing edges, overlapping edges, and overlapping vertices.

4 Base Algorithms for Hybridization

This section gives an explanation of the algorithms used for carrying out the hybridization of the proposed algorithm.

4.1 Univariate Marginal Estimation Algorithm (UMDA)

Introduced by Mühlenbein, [11] this is a particular case of EDAs which is considered as having no dependencies; its distribution of n-dimensional joint probability factorizes as a product of n univariate and independent probability distributions.
Example:

$$p_l(x) = \prod_{i=1}^{n} p_l(x_i) \tag{1}$$

The joint probability distribution of each generation was estimated from individuals $p_l(x)$ selected. The joint probability distribution factorizes as the product of independent univariate distributions.
Example

$$p_l(x) = p(x \mid D_{l-1}^{Se}) = \prod_{i=1}^{n} p_l(x_i) \tag{2}$$

Every univariate probability distribution is estimated by marginal frequencies:

$$p_l(x) = \frac{\sum_{j=1}^{N} d_j(X_i = x_i | D_{l-1}^{Se})}{N}$$

where: **(3)**

$$d_j(X_i = x_i | D_{l-1}^{Se}) = \begin{cases} 1 & \text{if on the } j\text{-th case of } D_{l-1}^{Se}, X_i = x_i \\ 0 & \text{in another case} \end{cases}$$

```
Pseudocode UMDA
$D_0$ Generate M individuals at random (initial population)
Repeat for l = 1, 2, . . . until stop criterion is verified.
$D_{l-1}^{Se} \leftarrow$ Select N $\leq$ M individuals from $D_{l-1}$ according to selection method.
$p_l(x) = p(x | D_{l-1}^{Se}) = \prod_{i=1}^{n} p_l(x_i) = \prod_{i=1}^{n} \frac{\sum_{j=1}^{N} d_j(X_i = x_i | D_{l-1}^{Se})}{N} \leftarrow$ Obtain estimate of combined probability distribution
$D_l$ Sample M individuals (new population) from $p_l(x)$.
```

4.2 Hill Climbing Algorithm

The hill climbing algorithm (HC) [15] is an optimization technique which belongs to the local search family. This algorithm uses a series of iterations where it is constantly shifting towards the direction with a better value. When the algorithm reaches a point where its result cannot be further improved, it needs to start all over at another point where it can direct its search. This is achieved by a random restart of the algorithm. This technique performs a series of climbing the top searches from randomly generated initial states. The best result obtained thus far is saved as the algorithm's iterations progress and it stops when no significant progress has been accomplished. The algorithm's stop condition may be set by a fixed number of iterations or when the best result has not improved over the course of a number of iterations.

```
Pseudocode HC
function HILL_CLIMBING(problem) returns a solution state
  inputs: problem, a problem
  static:   current, a node
            next, a node
  current ? MAKE-NODE(INITIAL-STATE[problem])
  loop do
```

```
        next ? a highest-valued successor of current
        if VALUE[next] ≤ VALUE[current] then return
        current ? neighbor
    end
```

5 Proposed hybrid HUx Algorithm

The hybrid HUx algorithm [6] [4] is based mainly on the UMDA algorithm; hence, it was necessary to modify this algorithm. Hybridization of the algorithm is achieved by applying 10 cycles of the HC algorithm with random restart to the best individual in the population. The HC algorithm was implemented by means of elitism because the algorithm was held back at a local optimum.

Hybridization performed in the UMDA algorithm took place as follows:

- Elitism was applied only if the best individual in the population had a better degree of adaptability to the problem (fewer crossings) as compared to the worst individual in the population. In this case, the best individual in the population is saved in the worst individual, thus rearranging the whole population.
- Hybridization was performed by applying 10 cycles of the HC with random restart to the best individual in the population. The result of this hybridization has the potential of improving the individual's degree of adaptability (which is what happens in most cases) or, it is possible for the degree of adaptability to worsen. Were this to be the case, convergence of the algorithm is achieved regardless because at the time of implementing elitism in the population, the best solution found thus far is never lost.

5.1 Pseudocode HUx

```
// for hybridization and elitism
public class  UMDA  {boolean HUx;
    public  UMDA(boolean HUx) {/* constructor for
                                  hybridization */
        this.HUx = HUx;
}
PopG oldpop, newpop;
    IndividualG best;
    //generate initial population
    For (i = 0; i < size_pob; i++)
        oldpop[i] = Individual;
    //random generation of vertices
    For (j = 0; j < total_vertex; j++)
        oldpop[i].vertex[j] = generateVertex;
    newpop = oldpop; //algorithm cycle
    //algorithm cycle for new population
    For (i = 0; i < max_generation; i++)
    {
        evaluateFobject;    /* evaluates population
                               quality */
        sortPop;           /* descending sort by total
                              cross */
        //initiates implementation of hybridization
        if this.HUx then
        {
            elitism;//elitism applied the population
            applyHC; /* 10 cycles of the HC applied to
best individuals*/
        // end of implementation hybridization
        best = saveBest;    /save the best individual
        calculaVectoProb;       /* estimated
                                  probability vector */
        oldpop = newpop;        /* copy new population
                                 in older population */
    }
}
```

6 Experiment Design

In order to compare the four algorithm implemented in this paper, seven graphs were used to perform experiments. The seven graphs were selected from the papers [14],[1], [3], [9], [13] to use them as benchmarks. Table 1 shows the general aspects of all graphs researched. This table shows the total number of vertices and edges in each graph as well as a density column, which is the ratio between existing edges in the graph and the maximum possible number of edges, which would be the complete graph *Kp*, having the same number of vertices *p* [14].

Table 1. General properties of graphs taken from literature.

Graph	Vertices (p)	Edges (q)	Density
Simple	8	12	0.429
Herschel	11	18	0.327
Hobbs	20	36	0.189
Tree	34	33	0.059
Star	17	40	0.294
Grid 4 x 4	16	24	0.200
Composite	40	69	0.088

The Simple graph was taken from [13] and it is the simplest of all graphs, having 8 vertices and 12 edges. The Herschel graph, taken from [1] has 11 vertices and 18 edges. The tree graph, taken from same paper, has 34 vertices and 33 edges and Star graph, also taken from the same paper has 17 vertices and 40 edges. The Hobbs graph was taken from the [9] and it has 20 vertices and 36 edges. The characteristic of this graph is that it is non-planar. The Grid graph was taken from [3]. This graph has 16 vertices and 24 edges and it corresponds to a 4 x 4 grid. The composite graph is the largest of all graphs, having a total of 40 vertices and 69 edges. This is a proper graph, which was created to test the hybrid HUx algorithm's stability before graphs with a greater number vertices and edges as described above. The composite graph [4] was built using vertices and edges from the Hobbs graph [9] as well as the Ebner graph [2]. The Ebner graph has a total of 20 vertices and 33 edges. This graph was taken from the literature and it was used only for building the composite graph.

All algorithms implemented in this paper were executed using the same number of iterations for all graphs taken from the literature, 3,600 evaluations in the case of the simple graph and 20,000 evaluations in all other instances. Additionally, 20 executions were performed independently for every individual algorithm. The average number of crossings found by the metaheuristics which were implemented demonstrates that the hybrid HUx algorithm outperformed all other algorithms.

The GA uses the following parameters; the reproduction cycle included 500 generations, a population composed by 40 individuals, a 90% crossover probability, and a 10% mutation probability. The above parameters permit the accomplishment of best results for this particular algorithm.

The algorithm HC with random restart also employs iterations (cycles) when searching for the best solution; therefore, we worked with 20,000 iterations. In this algorithm, one of the graph's vertices is selected at random and its coordinates are replaced by new vertex coordinates which are randomly generated in the plane.

The UMDA used 200 generations along with a population size containing 100 individuals, and a 50% truncation rate of the population. These parameters were the ones that yielded the best results for this particular algorithm.

The HUx algorithm uses a method of elitism in order to avoid losing the best solution found. It also uses 10 cycles of the up hill climber with random restart. This algorithm's parameters are as follows: a total of 200 generations, a population size containing 100 individuals, and a 50% truncation rate of the population. This is the top combination of parameters that yield the best results for this particular algorithm.

In the case of Composite graph, 800 generations were used, along with a population size containing 100 individuals, a 90% crossover probability and 10% mutation probability for population-based algorithms (GA, UMDA and HUx). In the case of the HC algorithm, 80,000 evaluations were performed (800 generations x 100 individuals) for finding the best solution. These parameters are the ones that yielded the best results for each algorithm.

In general, we used a total of 3,600, 20,000 and 80,000 evaluations (cycles) and a total number of 20 independent executions for each algorithm. This measurement intends to compare results obtained by our algorithms to those obtained from a group of papers included in the literature, in which graphs are drawn by means of general purpose methods, such as the Genetic Algorithm, the Hill Climbing algorithm and Univariate Marginal Estimation Algorithm. For instance, the graph designated as Simple [14] used 3,600 evaluations per execution. In order to compare it with the algorithms used in this paper, it was necessary to carry out 60 generations on 60 individuals for the population-based algorithms (GA, UMDA and HUx). This section lists all algorithms employed for completing the hybridization of the proposed algorithm.

7 Results and Discussion

The results of this experiment appear on tables 2, 3, 4, 5, and the discussions as follows.

7.1 Comparison of results among algorithms

Table 2 shows the overall average number of crossings found by each algorithm, as well as the overall average number of crossings found for each graph taken from the literature. This table shows how the hybrid HUx algorithm averages, for the most part, slightly less than 1 crossing per graph for the 20 executions carried out for each one, followed by the HC with random restart. Likewise, it can be observed that the hybrid HUx algorithm is the best of all algorithms with an overall average of 1.03 crossings per graph, while the worst one is the GA algorithm, whose overall average is 5.87 crossings per graph.

Table 2. Average number of crossings for graphs obtained from the literature.

	Average number of crossings found				
Graph	GA	UMDA	HUx	HC (random reboot)	Total
Simple	0.00	0.00	0.00	0.00	0.00
Herschel	1.70	2.05	0.40	0.75	1.23
Tree	7.70	1.05	0.25	1.20	2.55
Hoobs	11.35	9.90	4.80	6.15	8.05
Star	12.60	8.20	0.55	0.70	5.51
Grid	1.65	1.10	0.15	0.80	0.93
Total	5.83	3.72	1.03	1.60	

Further analyzing the results, one can see that the Hobbs graph [9] is the most difficult graph, having an overall average of 8.88 crossings found for all the algorithms implemented, while the Simple graph is the easiest of all, having an overall average of 0 crossings found for every algorithm implemented in this paper.

Table 3 shows the optimal frequency of occurrence reported in each of the graphs implemented in this paper. The frequency is defined by the number of times the algorithm reached the optimum. The table shows the optimal frequency of occurrence reached in the 20 independent evaluations that were executed for each algorithm.

Table 3. Number of times the algorithm found the optimum in a total of 20 evaluations.

Optimal frequency of occurrence				
Graph	GA	UMDA	HUx	HC (random reboot)
Simple	20	20	20	20
Herschel	8	2	16	14
Tree	0	7	15	8
Hoobs	0	0	6	3
Star	0	2	17	16
Grid	6	9	17	12
	28%	33%	76%	61%

Table 4 shows the optimal frequency of occurrence. The HUx algorithm outperforms all other algorithms, in some instances by more than 50%. The following table shows the percentage of occurrence of the optimum and it can be seen that the HUx algorithm has an effectiveness rate of 76% in obtained results, versus the 61% rate shown by the HC algorithm with random reboots. This is a 15% difference with regard to the effectiveness rate as compared to the second best algorithm.

Table 4. Average optimal frequency of occurrence in a total of 20 evaluations.

Average Frequency of occurrence of the optimum				
Graph	GA	UMDA	HUx	HC (random reboot)
Simple	100%	100%	100%	100%
Herschel	40%	10%	80%	70%
Tree	0%	35%	75%	40%
Hoobs	0%	0%	30%	15%
Star	0%	10%	85%	80%
Grid	30%	45%	85%	60%
	28%	33%	76%	61%

Table 4 shows that the percentage of occurrence of the optimal HUx algorithm lies within an effectiveness rate of 100% and 75% in the case of the Simple graph, the Herschel graph, the Tree graph, the Star graph and Grid graph. Only the Hobbs graph, which is the most difficult of all, a 30% effectiveness rate was obtained. Nonetheless, it still exceeded the 50% effectiveness rate shown by the HC algorithm with random reboot, which is the second best of all algorithms.

Finally, it is worth mentioning that the reach of our objective, which in this case entailed a drawing having the minimum number of crossings in the graph's edges, was fully achieved by the hybrid HUx algorithm as well as by the HC algorithm with random reboot. The search for the optimal solution for all graphs taken from the literature includes zero crossings on the graph's edges; only in the Hobbs graph which is non-planar, in this case the optimal solution is 2 crossings on the edges of this graph.

Table 5 shows the reach of goals accomplished by the different metaheuristics for each of the graphs taken from the literature.

Tabla 5. Reach of the goals obtained by the different metaheuristics.

	Scope of Objectives			
Graph	GA	UMDA	HUx	HC (random reboot)
Simple	0	0	0	0
Herschel	0	0	0	0
Tree	2	0	0	0
Hoobs	5	3	2	2
Star	2	0	0	0
Grid	0	0	0	0

When analyzing the results obtained by the UMDA algorithm and the HC algorithm with random reboot, it is obvious that the reach of their objectives is very similar, taking into account that this feature makes the decision to implement a new algorithm having the virtues of these two algorithms in order to overcome the results possessed these two algorithms up to that moment. This is how the hybrid HUx algorithm is implemented.

7.2 Comparison of HUx algorithm with literature results

A comparison of the results obtained by the hybrid HUx algorithm and some of the algorithms reported in the literature comprising part of the state of the art on graph drawing for the minimization of crossings on their edges is shown below. The way results are presented in this section differs from the way results were presented in the previous index because most of the papers herein cited does not allow a direct comparison given the fact that only in a few instances are statistical summaries provided. Hence, the authors only give a description and show some of the results obtained. Therefore, the way of comparing the results for each graph is done separately.

The Simple graph for the paper presented in [13], yields 93% of drawings with no crossing after 3,600 evaluations (60 generations x 60 individuals). With regard to the same graph presented by [14], the Stochastic Hill Climbing (SHC), yields a 100% drawings with no crossings after 550 evaluations, while the hybrid HUx algorithm yields 100% of drawings with no crossings after 252 evaluations on average in a total of 20 independent executions. This is a better result since it uses only half of the evaluations to achieve the goal with respect to the SHC.

The paper of [99] yields a solution with four crossings after 20,000 evaluations (1000 generations x 20 individuals). The SHC [14], yields two solutions with three crossings, four solutions with four crossings and three solutions with five crossings during the 20 executions, and no details are given with regard to number of evaluations needed to accomplish the solution. The hybrid HUx algorithm presents six solutions with two crossings, three solutions with three crossings, two solutions with four crossings and two solutions with five crossings after 9,034 evaluations on average during the 20 executions of the algorithm. These results are by far superior to the those obtained by Hobbs and Rossete, since six solutions were obtained with two crossings, something not achieved by the SHC algorithm.

The results presented by [1] for the tree graph yields only one solution with no crossings. The SHC [14], yielded ten solutions with no crossings and six solutions with one crossing after 3,000 evaluations. The hybrid HUx algorithm yielded fifteen solutions with zero crossings and five solutions with one crossing in just 4,841 evaluations on average in the twenty executions. These results also favor the hybrid HUx algorithm since it yielded five more solutions with zero crossings than the SHC algorithm.

For the Star graph [1], authors obtained one solution with no crossings. The SHC [14] yielded five solutions with no crossings and two solutions with one cross during the 20 executions, no details are given with regard to the number of evaluations required to attain these results. The hybrid HUx algorithm yielded seventeen solutions with zero crossings, one solution with one crossing, one solution with three crossings and one solution with seven crossings upon completing 4909 evaluations on average during the 20 executions for the algorithm. The results obtained for this graph are also much higher since the HUx yielded twelve results with zero crossings more than SHC algorithm.

The Herschel graph SHC [14], yielded seventeen solutions with no crossings after 3,000 evaluations during the 20 executions. The authors of the paper [1] only obtained one solution with no crossings when using the mutation based on the springs algorithm. However, when it is not used, it is not possible to obtain one single solution with no crossings. The hybrid HUx algorithm yielded 16 solutions with no crossings after 4,194 evaluations on average and four solutions with two crossings after 20 executions of the algorithm. For this graph, results obtained by the SHC and hybrid HUx algorithm are very similar, although the SHC obtained included only 1 more crossing than the hybrid HUx algorithm.

The Grid graph [3], yields a solution with no crossings upon completion of 50,000 evaluations (5000 generations x 10 individuals). In the paper of [14], the SHC yielded 12 solutions with no crossings after 3,000 evaluations during the 20 executions of the algorithm, while the hybrid HUx algorithm yielded seventeen solutions with zero crossings and three solutions with one crossing after 4,018 evaluations on average during the 20 executions of the algorithm. The results again favored the hybrid HUx algorithm since it yielded five crossings more than the SHC.

A larger graph was designed with more vertices and edges than graphs taken from the literature, This graph include the vertices and edges of two graphs; namely, the Hobbs graph which has 20 vertices and 36 edges, and the Ebner graph, which has 20 vertices and 33 edges. The graph resulting from the combination of these two graphs was named “Composite graph”. This union yielded the Composite graph, which was built having 40 vertices and 69 edges, which is the largest graph of all. The results of the composite graph are shown below in Table 6.

Table 6. Average optimal frequency of occurrence in a total of 20 evaluations.

	Composite graph			
Algorithm	Average Crossings	Objective Scope least 3 crossings	Frequency of occurrence	Average frequency of occurrence
GA	63	35	0	0%
UMDA	34	19	0	0%
HUx	11	3	1	5%
HC (reboot)	10	5	0	0%
	29.542	15.50	0.25	1%

The table's first column shows that the average number of crossings is 11 for the hybrid HUx algorithm, while the HC algorithm with restart obtained a total of 10 crossings, which is one less than the hybrid HUx algorithm, making it the best of all algorithms; however, this result is very similar to that obtained by the hybrid HUx algorithm. Furthermore, it is known that the minimum number of crossings of the Composite graph is 3 because the minimum number of crossings for the Ebner graph is 1, and the minimum number of crossings for the Hobbs graph is 2, as shown in Table 6. By looking at column two on the Table 6, it can be observed that the only algorithm that achieved the objective is the hybrid HUx algorithm, as shown by columns three and four on Table 6. Here one can observe that the only algorithm that contributing to the statistical data is the hybrid HUx algorithm.

Figure 1 shows the Composite graph drawing, which was minimized by the hybrid HUx algorithm. The HUx was the only algorithm that reached the three crossings objective. The Minx System displays the Composite graph [4].

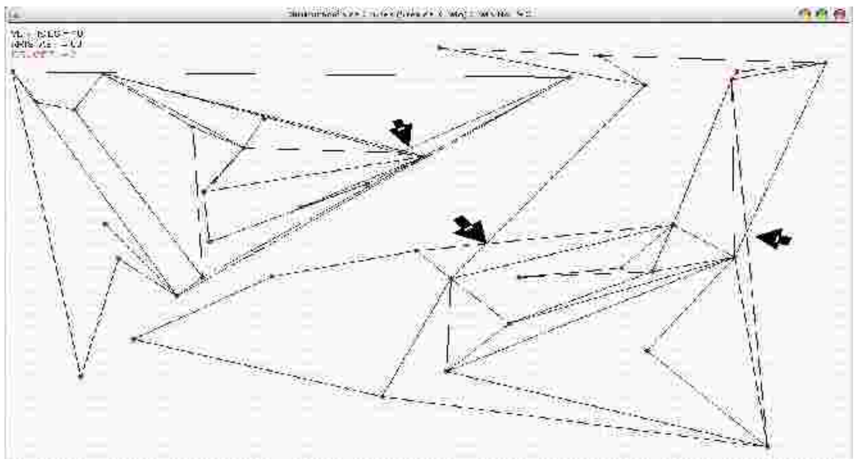

Fig. 1 Composite graph displayed by the system Minx.

8 Conclusions

Among the main contributions generated by this research paper, the creation of the hybrid HUx algorithm that is used in the graph drawing crossing minimization of edges problem efficiently is included. This can be used in automatic graph drawing.

It was demonstrated that this tool was implemented successfully in all experiments carried out; therefore, it is an efficient way to solve the problem of minimizing the crossings of the edges of a graph. Furthermore, analysis results showed that this tool exceeded the quality of results produced by the various metaheuristics implemented in this research paper as well as the results obtained by other metaheuristics taken from the literature and that make up part of the state of the art.

The design, development and deployment of the hybrid HUx metaheuristic, has been of paramount importance as it leverages the most outstanding properties of two metaheuristics such as the UMDA which is a global search metaheuristic and the HC algorithm is an algorithm of local search. UMDA characteristics to capture global properties of the solutions by estimating a probabilistic model (independence model) and secondly the speed and efficiency shown by the HC algorithm makes the hybrid HUx algorithm a tool most effective in the search for minimization of crossings on the edges of a graph.

The Minx system reported in [4] provides users a tool for a friendly and easy to use graphs display. The automatic drawing of minimized graphs makes it easier for the user to compare results appearing in separate windows, giving the user the opportunity to choose the graph design which best suits their needs.

References

1. Branke, J., Bucher, F., Schmeck, H.: A genetic Algorithm for drawing undirected graphs, Proceedings of 3rd Nordic Workshop on Genetics Algorithms and their Applications, Alander, J. T. (Ed.), pp. 193-206 (1997).
2. Ebner, D.: Optimal Crossing Minimization Using Integer Linear Programming, Unity Technology of Vienna, 87 pages (2005).
3. Eleoranta T., Mäkinen E.: A genetic Algorithm for Drawing Undirected Graphs, Departament of Computer and Information Science, University of Tampere Finland, Work supported by the Academy of Finland (Project 35025) (2001).
4. Enríquez S.: Metaheurística Evolutiva Híbrida para la Minimización de Cruces de las Aristas en el Dibujado de Grafos, Universidad Autónoma de Aguascalientes, Centro de Ciencias Básicas, Tesis y Disertaciones Académicas, page 91 (2009).
5. Enríquez S., Ponce de León E., Díaz E.: Calibración de un Algoritmo Genético para el Problema de la Minimización de Cruces en las Aristas de un Grafo, Avances en Computación Evolutiva, Memorias del IV Congreso Mexicano de Computación Evolutiva, Centro de Investigación en Matemáticas (CIMAT), pp. 61-66 (2008).
6. Enríquez S., Ponce de León E., Díaz E., Padilla A.: Sistema Minx para el Dibujado de Grafos con el Menor Número de Cruces en sus Aristas, Memorias del Cuarto Congreso Estatal la Investigación en el Posgrado, Universidad autónoma de Aguascalientes, http://posgrado.uaa.mx/posgrado/, page 3 (2008).
7. Garey M. R., Johnson D. S.: Crossing number is NP-complete. SIAM Journal on Algebraic and Discrete Methods, vol. 4, pp. 312-316 (1983).

8. Herman I., Melancon G., Marshall S.: Graph Visualization and Navigation in Information Visualization: A Survey, Visualization and Computer Graphics, IEEE Transactions, vol. 6, Issue: 1, pp. 24-43 (2000).
9. Hobbs, M. H. W., Rodgers P. J.: Representing Space: A Hybrid Genetic Algorithm for Aesthetic Graph Layout, Frontiers in Evolutionary Algorithms, FEA'98, Proceedings of 4th Joint Conference on Information Sciences, JCIS'98, vol. 2, pp. 415-418 (1998).
10. Larrañaga P., Lozano J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, 382 pages (2002).
11. Mühlenbein H., Mahnig T., Ochoa A.: Schemata Distributions and Graphical Models in Evolutionary Optimization, Journal of Heuristic, Vol. 5, No. 2, pp. 215-247 (1998).
12. Pach J., Tóth G.: Which crossing number is it, anyway? Proceedings of the 39th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Washington, DC, USA, pp. 617-626 (1998).
13. Rossete A., Ochoa A.: Genetic Graph Drawing, Proceedings of 13th International Conference of Applications of Artificial Intelligence in Engineering, AIENG'98, Adey, R. A., Rzevski, G., Nolan, P. (Ed.) Galway, Computational Mechanics Publications, pp. 37-40 (1998).
14. Rossete A.: Un Enfoque General y Flexible para el Trazado de Grafos, Tesis presentada en opción al grado científico de doctor en Ciencias Técnica, Facultad de Ingeniería Industrial , CEIS, La Habana, Cuba (2000).
15. Russell S., Norving P.: Artificial Intelligence : A Modern Approach, A Simon & Schuster Company Englewood Cliffs, New Jersey, Prentice Hall, pp. 111-114 (1995).