Usage of Genetic Algorithm for Lattice Drawing

Suhail Owais, Petr Gajdoš and Václav Snášel

Department of Computer Science,
VŠB - Technical University of Ostrava,
tř. 17. listopadu 15, 708 33 Ostrava-Poruba
Czech Republic
Suhail.Owais@vsb.cz
Petr.Gajdos@vsb.cz
Vaclav.Snasel@vsb.cz

Abstract. Lattice diagrams, known as Hasse diagrams, have played an ever increasing role in lattice theory and fields that use lattices as a tool. Initially regarded with suspicion, they now play an important role in both pure lattice theory and in data representation. Using evolutionary algorithms-genetic algorithms for drawing a lattice diagram is the main goal of our research. It depends on stochastic methods on searching of concepts' positions, with trying to draw it with less number of intersections.

Keywords: FCA, Genetic Algorithm, Hasse Diagram

1 Graphic representation of concept lattice

These Hasse diagrams are an important tool for researchers in lattice theory and ordered set theory and are now used to visualize data. This paper also shows a new approach how to draw a Hasse diagrams using genetic algorithm.

In the following paragraphs we would like to mention known approaches of lattice drawing and main points which make a Hasse diagrams or random graphs more readable. After that we explain our approach and all important terms from the area of genetic algorithms.

An ordered set $P = (P, \leq)$ consists of a set P and a partial order relation $\leq$ on P. That is, the relation $\leq$ is reflexive ($x \leq x$), transitive ($x \leq y$ *and* $y \leq z \Rightarrow x \leq z$) and antisymmetric ($x \leq y$ *and* $y \leq x \Rightarrow x = y$). If P is finite there is a unique smallest relation $\prec$, known as the cover or *neighbour* relation, whose transitive, reflexive closure is $\leq$. (Graph theorists call this the transitive reduct of $\leq$.) A *Hasse* diagram of P is a diagram of the acyclic graph $(P, \prec)$ where the edges are straight line segments and, if $a < b$ in P, then the vertical coordinate for a is less than the one for b. Because of this second condition arrows are omitted from the edges in the diagram.

A *lattice* is an ordered set in which every pair of elements a and b has a least upper bound, $a \vee b$, and a greatest lower bound, $a \wedge b$, and so also has a *Hasse* diagram.

Radim Bělohlávek, Václav Snášel (Eds.): CLA 2005, pp. 82–91, ISBN 80–248–0863–3.

1.1 Known approaches of lattice drawing

By the 1970s diagrams had become an important tool in lattice theory. There exist some known methods how to draw the Hasse diagrams. The first one it called *Hierarchical Layout* Partially ordered sets, for the purposes of layout, are seen as hierarchies. An annotated bibliography by Battista et. [1] describes the approach as

> A hierarchical drawing of an
> acyclic diagram is an upwards
> polyline drawing where the vertices
> and bends are constrained
> to lie on a set of equally spaced
> horizontal lines.

Sugiyamas algorithm is typical of a variety of algorithms proposed for the layout of such diagrams. There are three main steps of whole process.

1. Partition the vertices into layers.
2. Reduce the number of edge crossings by swapping the horizontal ordering of vertices.
3. Minimize the number of bends, where bends are dummy nodes inserted to ensure that lines travel only between adjacent layers.

Next there are additive or non additive diagrams and the methods to draw Hasse diagrams which were introduced by Rudolf Willw [11]. But we can mention the main points of each drawing.

These are the most important points to draw a readable graph:

- Avoid crossing of lines.
- Try to draw parallel lines.
- Identify known structures: cubes, rectangles ...
- Layer the nodes: draw nodes on the same layer, if their concept's extents have the same size.
- Try to draw steep lines, avoid flat ones.

2 Evolutionary Algorithms

Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution, which applies the principles of evolution found in nature to the problem of finding an optimal solution to a solver problem. An evolutionary algorithm is a generic term used to indicate any population-based optimization algorithm that uses mechanisms inspired by biological evolution, such as reproduction, mutation and recombination. Candidate solutions to the optimization problem play the role of individuals in a population, and the cost function determines the environment within which the solutions "live" (see figure

1). Evolution of the population then takes place after the repeated application of the above operators. Figure 1 shows the structure of a simple evolutionary algorithm for solving a problem. Genetic algorithm is the most popular type of evolutionary algorithms.

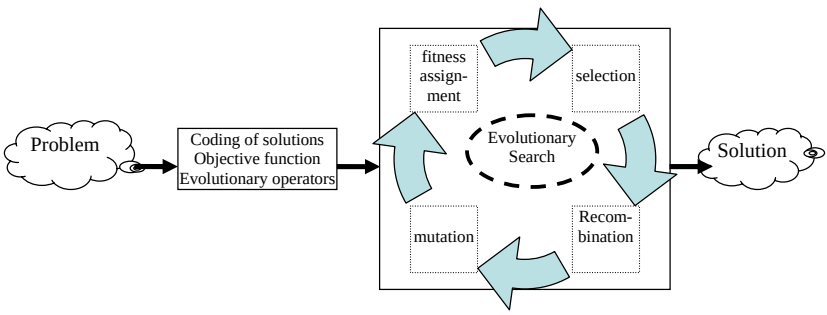

Fig. 1. Basic scheme of our solution.

3 Genetic Algorithms (GA)

Genetic algorithms (GA) described by John Holland in 1960s and further developed by Holland and his students and colleagues at the University of Michigan in the 1960s and 1970s. GA used Darwinian Evolution to extract nature optimization strategies that use them successfully and transform them for application in mathematical optimization theory to find the global optimum in defined phase space [4][5][6].

GA is used to extract approximate solutions for problems through a set of operations "fitness function, selection, crossover, and mutation". Such operators are principles of evolutionary biology applied to computer science. GA search process depends on different mechanisms such as adaptive methods, stochastic search methods, and use probability for search.

Using GA for solving most difficult problems that searches for accepted solution; where this solution may not be the best and the optimal one for the problem. GA are useful for solving real and difficult problems, adaptive and optimization problems, and for modeling the natural system that inspired design [9][2].

Some applications that can be solved by GA are: Scheduling 'Job, and Transportation Scheduling', Design 'Communication Network design', Control 'Gas pipeline control', Machine Learning 'Designing Neural Networks', Robotics 'Path planning', Combinatorial Optimization 'TSP, Graph Bisection, and Routing', Signal Processing 'Filter Design', Image Processing 'Pattern recognition', Business 'Evaluating credit risks', and in Medical 'Studying health risks for a population exposed to toxins' [10].

4 Genetic Algorithms Operators

Typically, any genetic algorithm used for purpose of optimization consists of the following features:

1. Chromosome or individual representation.
2. Objective function "fitness function".
3. Genetic operators (selection, crossover and mutation).

Where applying GA done over a population of individuals or chromosomes shows that several operators are utilized. Presenting of GA process described in a flowchart shown in 2.

4.1 Chromosome Encoding

Encoding by simple; representation of integer numbers by binary encoding. Representation for chromosome (suggested solution) will be in a string of symbols to simplify using of GA operators and to help in reproduction of optimal solution. Encoding may be in string of bits, or in arrays, or in lists, or in tree structure, and may be in other types. Some examples were shown in table 1.

Encoding Type	**Example**
Binary Encoding	1101100100110110
Permutation Encoding	8 5 6 7 2 3 1 4 9
Value Encoding	(back), (back), (right), (left)

Table 1. Examples for encoding types

4.2 Objective Function

The goal of GA is to find a solution to a complex optimization problem, which optimal or near-optimal. GA searches for better performing candidates, where performance can be measured in terms of objective"fitness" function. The objective is to choose the proportion of financial assets to hold in a portfolio such that risk is minimized given the constraint of achieving a specified level of return. Fitness function does as a metrics to measure scheduler performance for each chromosome in the problem.

4.3 Selection Operator

Selection determines which solution candidates are allowed to participate in crossover and undergo possible mutation. Select two chromosomes with the highest quality values from the population using one of the selection methods. Some of these methods are: *Elitism Selection, Roulette Wheel Selection, Tournament Selection, Rank Selection, and Stochastic Selection.*

4.4 Crossover operator

Promising candidates, as represented by relatively better performing solutions, are combined through a process of binary recombination referred to as crossover. This ensures that the search process is not random but rather that it is consciously directed into promising regions of the solution space. Crossover exchanges subparts of the selected chromosomes, where the position of the subparts selected randomly to produce offsprings. Some of crossover methods types are: *Single Point Crossover, Two Points Crossover, Multipoint Crossover, Uniform Crossover, and Arithmetic Crossover.*

4.5 Mutation operator

New genetic material can be introduced into the population through mutation. This increases the diversity in the population. Mutation occurs by randomly selecting particular elements in a particular offsprings.

GA process terminated by satisfies one of terminating conditions often include:

- Fixed number of generations reached.
- Budgeting: allocated computation time/money used up.
- An individual is found that satisfies minimum criteria.
- The highest ranking individual's fitness is reaching or has reached a plateau such that successive iterations are not producing better results anymore.
- Manual inspection. May require start-and-stop ability.
- Combinations of the above.

5 Genetic algorithm implementation

In the this chapter we try to describe our approach to draw a Hasse diagram for a concept lattice using genetic algorithm. There are five levels and ten concepts (nodes) in the figure 3. In the first generation the population of a chromosomes were generated randomly. At the beginning the number of levels is computed according to an incremental algorithm made by Petko Valtchev. Then we compute the *level position* for each node, some nodes may appear in different levels, e.g. the concept number six could be moved one level up. Next vector (see table 2) consists of all edges in the Hasse diagram. The numbers represent the indexes of nodes (concepts).

From	0	0	0	5	5	8	8	6	6	9	9	7	4	2
To	6	5	9	3	4	3	9	4	2	7	2	1	1	1

Table 2. Vector of diagram edges

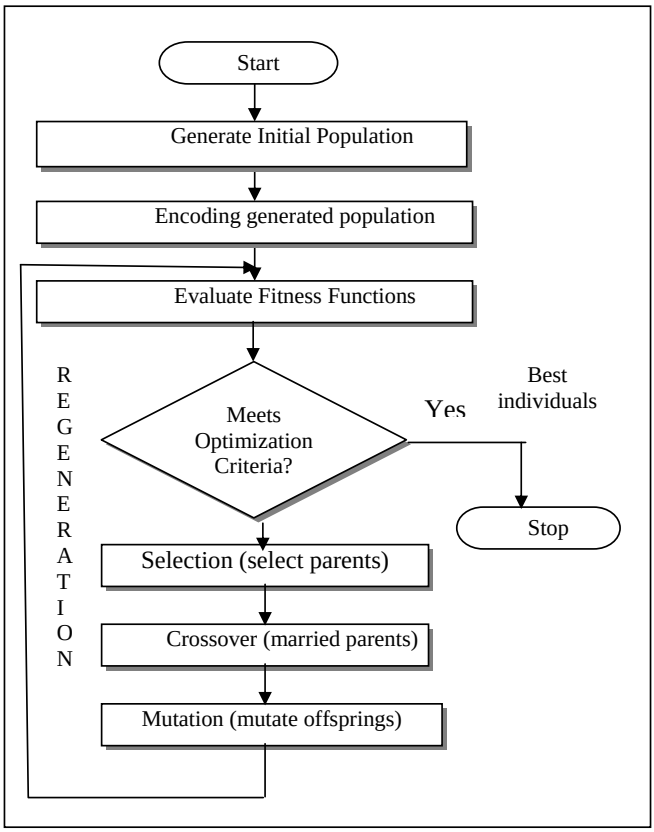

Fig. 2. asdadadad

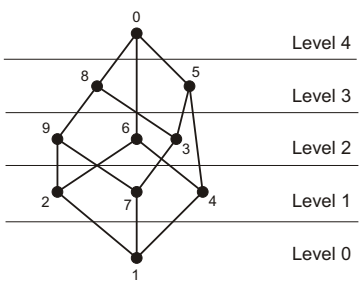

Fig. 3. Hasse digram with marked levels.

5.1 Chromosome encoding

Each chromosome was encoded as a vector which consists of x and y coordinates for all nodes in the graph. The coordinates will be generated randomly

for node with respect to the levels of nodes. Table 3 shows an example for such chromosome.

X-coord.	200	200	50	250	300	270	180	200	100	70
Y-coord.	50	400	300	200	300	100	200	300	100	200

Table 3. Chromosome encoding

6 Fitness function

We are looking for more readable graph. The most important constrains we look for the minimal number of edges' intersections.

6.1 Selection method

The best two chromosomes will be selected depends on the minimal number of intersections and called *parents*. They will be used in the other operators *crossover* and *mutation* to produce two new offspring.

6.2 Crossover

Three types of crossover will be implemented (*single point*, *two points* and *multi points* crossover) over selected parents. Following figures show examples for the three types.

For the single point crossover (see figure 4) the one node position will be randomly chosen. It will divide the parents' chromosomes into two parts. Then first offspring's chromosome consists of the first part from the first parent and second part from the second parent. The second offspring consists of the first part from the second parent and the second part from the first parent.

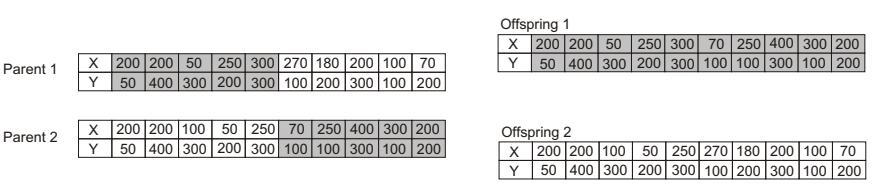

Fig. 4. Single point crossover.

For the two points crossover (see figure 5) two nodes' positions will be randomly chosen. This will divide the parents' chromosomes into three parts. Then

first offspring's chromosome consists of the first and third parts from the first parent and second part from the second parent. The second offspring consists of the second part from the first parent and the first and third parts from the second parent.

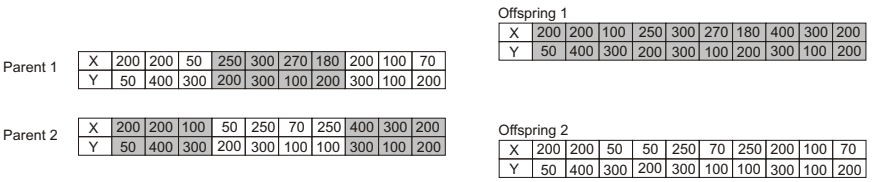

Fig. 5. Two points crossover.

For the multi points crossover (see figure 6) random number n of nodes will be chosen. Also we generate n number of nodes' positions randomly to swap these nodes to produce two new offspring.

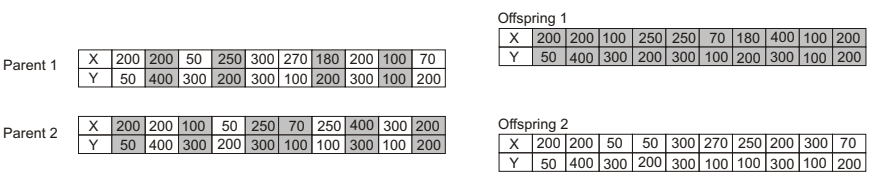

Fig. 6. Multi points crossover.

6.3 Mutation

The two new offspring will be mutated randomly to change some x coordinates depends on *mutation value*. For each node the random value will be selected and if this value is less then *mutation value* then new random number will be generated to define x coordinate of the node. Figure 7 shows an example. The x coordinate of the node number 4 was mutated from the value 250 to the 220 because the random number 0.1 for this node is less then the *mutation value*0.2. The Nodes number zero and one will be not mutated, because they will be placed in the middle of the $x - axes$ (see figure 7). Some nodes may be mutated by change its level ($y - coordinate$) depends on the position of parent concept and child concept. If there is a free level between selected concept and its parent concept, then it can be mutated. Correspondingly, if there is a free level between selected concept and its child concept. See figure 8.

then fitness values will be computed for the new offspring. These offspring may be replaced by the worst chromosome in the population if its fitness values exceed the fitness value of the worst chromosome. Now the new population is ready for a new generation.

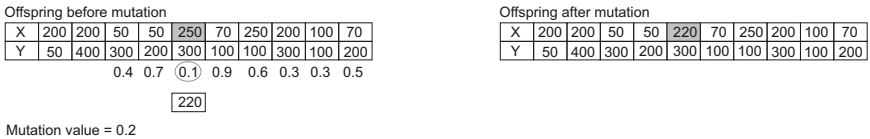

Fig. 7. Mutation of x-coordinates of nodes.

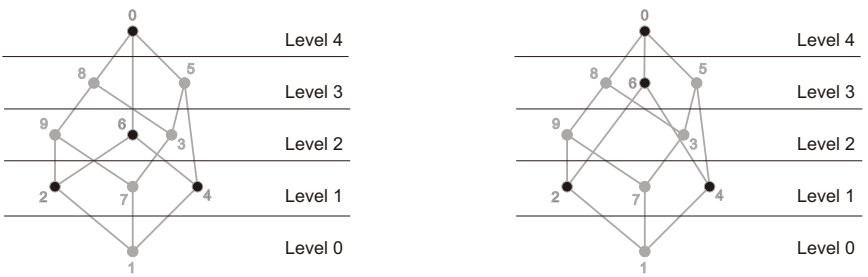

Fig. 8. Mutation of y-coordinate (level) of the node 6.

Also we explained how to get a good placing of nodes (concepts) in a graph with a minimal number of intersections. In our approach the main sence of fitness function is to obtain this number. But the function could be modified and other criteria could be added. For example a minimal distance between nodes, a graph symmetry or minimal sum of lengths of graph edges. But this is the point of concrete implementation.

6.4 Termination process

The process will be terminated if any of these cases:

1. there will be a chromosome with no intersection which means the best solution,
2. after processing limited number of generations,
3. if there is no improvement over the population in limited number of generations.

7 Conclusion

The genetic algorithms are used to solve hard problems, where the drawing of a nice and readable Hasse diagram is one of them. We introduced one possible way to draw it using genetic algorithm. Using different types of the crossover operators may produce god drawing results. Moving nodes from level to level by mutation operator may give us better results.

In our future work we will try to create an application for lattice drawing based on our approach. The shape of the graph could be improved using other characteristics like distances between nodes on the same level.

References

1. G. Battista, P. Eades, R. Tamassia, and I. Tollis. Algorithms for drawing graphs, an annotated bibiliography. *Computational Geometry, Theory and Applications*, 4:235–282, 1994.
2. H. Fang, P. Ross, and D. Corne. Genetic algorithms for timetabling and scheduling. http://www.asap.cs.nott.ac.uk/ASAP/ttg/resources.html, 1994.
3. B. Ganter and R. Wille. *Formal Concept Analysis.* Springer-Verlag Berlin Heidelberg, 1999.
4. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.
5. M. Melanie. *An Introduction to Genetic Algorithms. A Bradford Book.* MIT-Press, 1999.
6. M. Melanie and S. F. Genetic algorithms and artificial life. Santa Fe Institute, working Paper 93-11-072, 1994.
7. S. S. J. Owais. Timetabling of lectures in the information technology college at al al-bayt university using genetic algorithms. Master's thesis, Al al-Bayt University, Jordan, 2003. in Arabic.
8. H. Pohlheim. Evolutionary algorithms: Overview, methods and operators. GEATbx version 3.5., 2004. http://www.geatbx.com/docu/index.html.
9. E. P. K. Tsang and T. Warwick. Applying genetic algorithms to constraints satisfaction optimization problems. In *Proc. Of 9th European Conf. on AI.* Aiello L.C., 1990.
10. R. L. Wainwright. Introduction to genetic algorithms theory and applications. The Seventh Oklahoma Symposium on Artificial Intelligence, November 1993.
11. R. Wille. Lattices in data analysis: How to draw them with a computer. *Algorithms and Order*, pages 33–58, 1989.