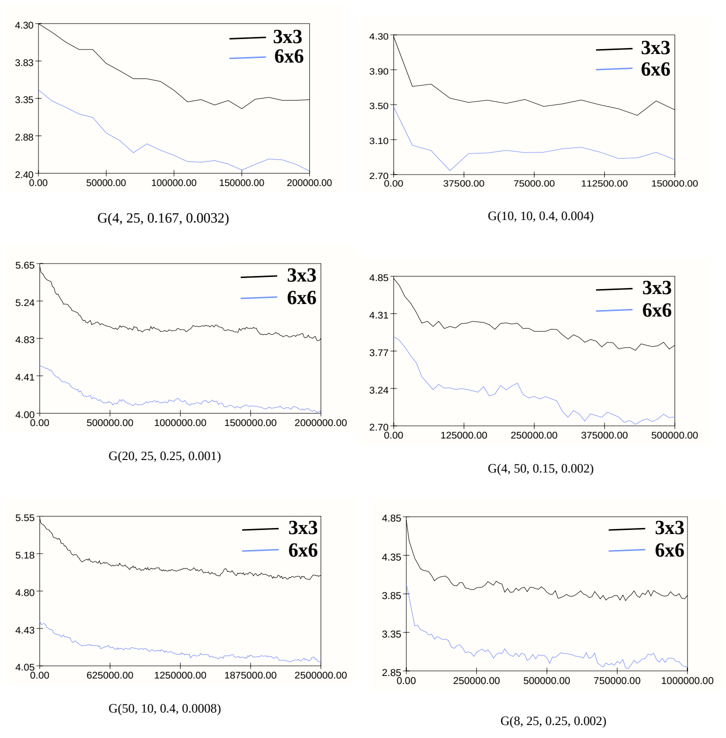


Figure 5: Variations of spatial entropy for different random graphs G_{Gar}

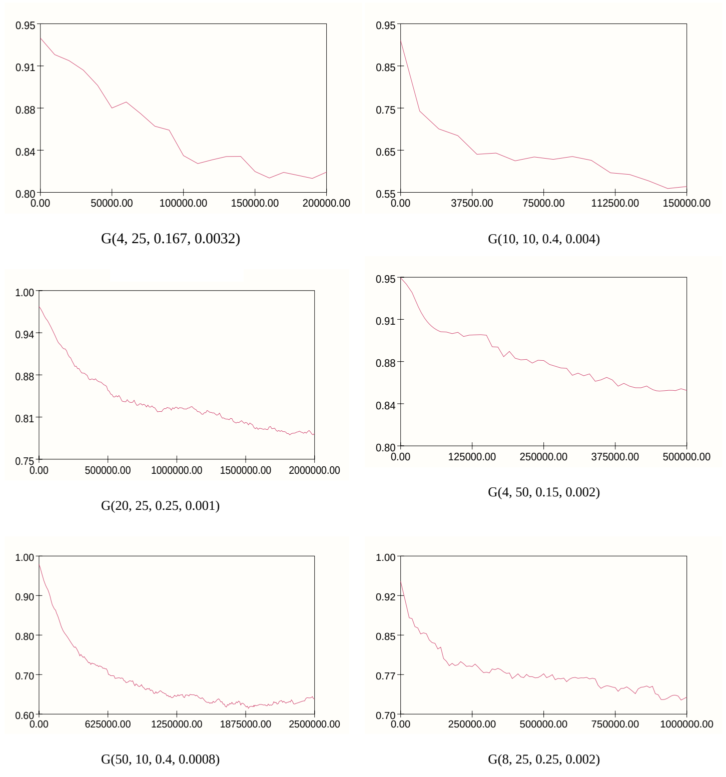
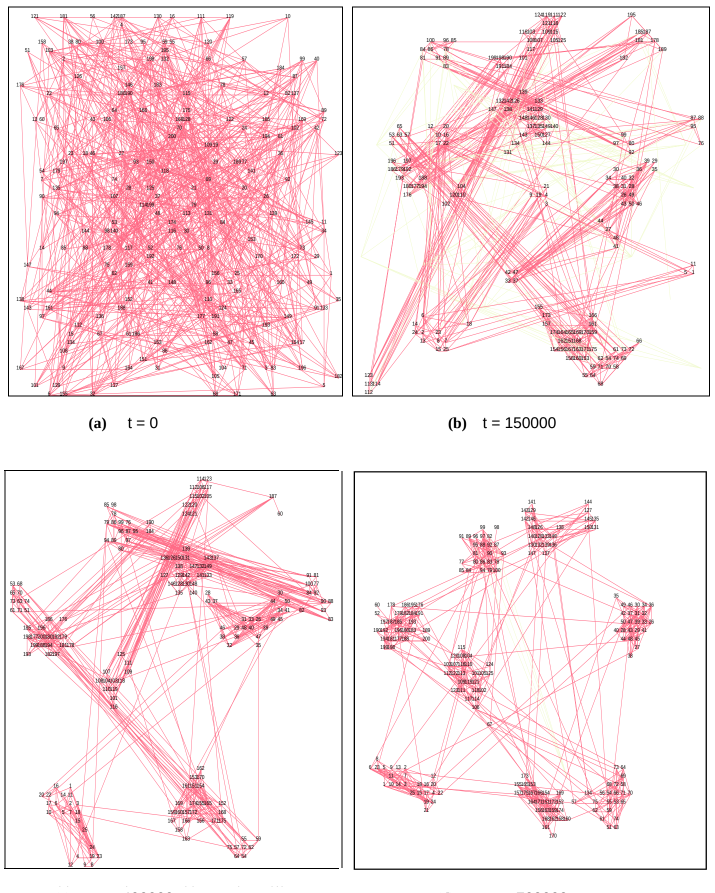


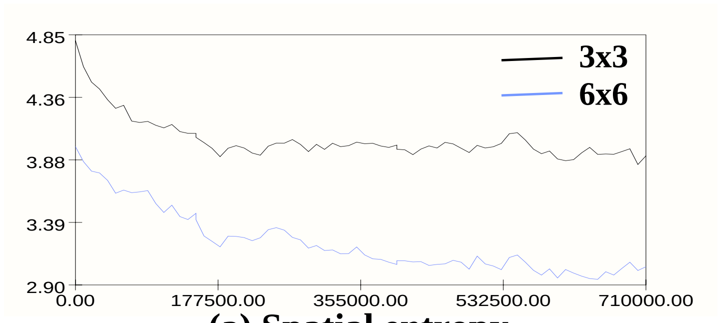
Figure 6: Variations of local distribution of d for different random graphs G_{Gar}


(a) t = 0 **(b)** t = 150000

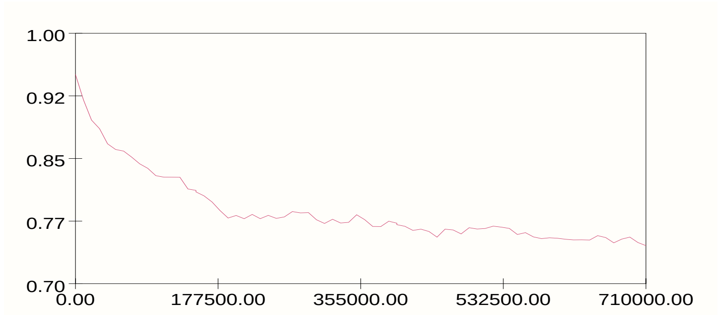
(c) t = 400000 **(d)** t = 700000

Figure 6. Graph $G_{GAR}(8, 25, 0.25, 0.0032)$ representation

Figure 4: Graph $G_{GAR}(8, 25, 0.25, 0.0032)$ representation

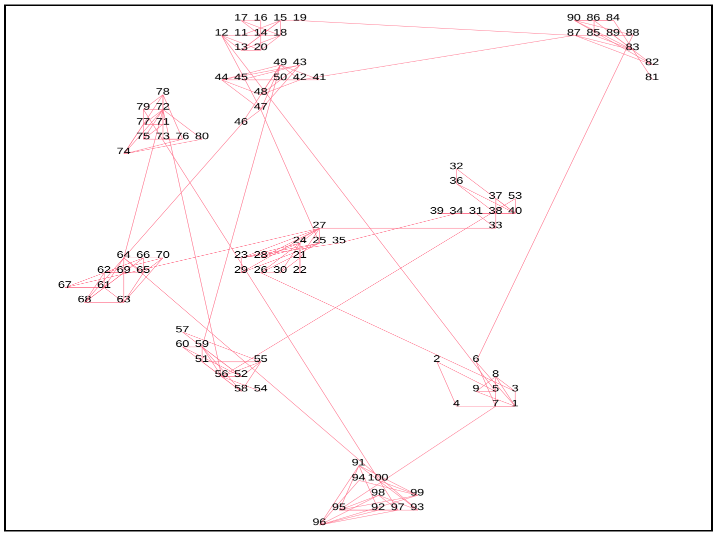


(a) Spatial entropy

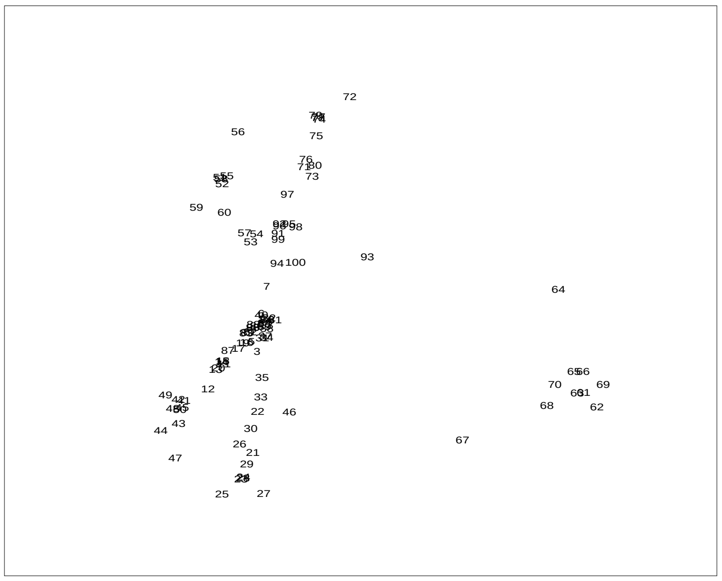


(b) Local distribution of dissimilarity

Figure 3: Graph $G_{GAR}(8, 25, 0.25, 0.0032)$



(a)



(b)

Figure 2: Graph $G_{GAR}(10, 10, 0.4, 0.004)$

to the same class, resulting in one cluster per expected class, and we arrive at the desired representation (Figure 4(d)).

3.3.2 Local Distribution of Dissimilarity Values

While the spatial entropy measure reflects an organisation of the vertices into clusters, it does not indicate whether vertices are placed into the correct classes or not. Using the class of random class described above, we know a priori the class size and to which class each vertex is expected to belong. The errors of allocation due to the probability pint are small enough as shown by Garbers and al. (1990) to be ignored here. To verify the good repartition of vertices into classes we study the evolution of the sum for each vertex vi of the dissimilarities between v_i and its nearest neighbours -for the l_1-distance- on the grid

$$\frac{1}{K_t} \sum_{v_i \in \Gamma} \sum_{v_i \in N_t(v_i)} d\left(v_i, v_j\right)$$

with $N_t(v_i)$ the set of the $(nc - 1)$ nearest neighbours of v_i minus the number of vertices of its class which are being displaced by automata at t. The sum is normalised by dividing by the total number K_t of dissimilarity calculations. The minimum value is reached when the nc closest cousins to every vertex all belong to the same class. Hence this measure reflects not only that vertices are placed in the"correct" cluster, but also whether cluster separation is good. Figure 3(b) illustrates its variation with time for the graph described above : the rapid initial clustering corresponds to the curve's large initial gradient, the subsequent agglomeration of the numerous small clusters then corresponding to a slower rate of decrease.

Figure 5 and 6 give both the variation of the spatial entropy and the local distribution of dissimilarity values for several graphs $G_{GAR}(k, nc, p_{int}, p_{ext})$ described in 4.2. ; the curves confirm the good visual results noticed previously. In order to compare with a family of graph well-known in VLSI literature we only presented here experiments with classes of equal sizes. Nevertheless the algorithm was also applied to graphs containing a priori various vertex classes and the results were as good as for equivalent cardinalities (Layzell 1995). The only problem appears for graphs containing one or more big subgraphs with a lot of links ; instead of grouping all the vertices of such a set in a same class it tends to create different smallest classes.

4 Discussion

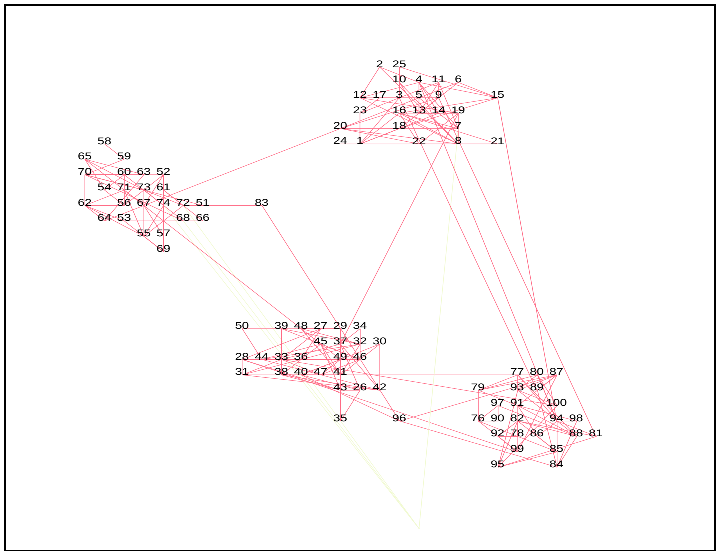


Figure 1: Graph $G_{GAR}(4, 25, 0.167, 0.0032)$ representation

3.3.1 Spatial entropy

Let $\Gamma_1, ..., \Gamma_r$ be a partition of the grid Γ in r sub-grids of same size (usually called grain from physics). The spatial entropy associated with this partition is defined by

$$-\sum_{i=1}^{r} p_t(\Gamma_i) log(p_t(\Gamma_i))$$

with $p_t(G_i)$ the proportion of vertices on G_i at time t. This measure reaches its minimum value 0 when their exists a sub-grid Γ_i which contains all the vertices and its maximum value 1 in case of uniform distribution $pt(\Gamma_1) = ... = pt(\Gamma_r) = 1/r$. It is used to reveal the existence of a structuration of the vertices on the space : the evolution from the initial random distribution of vertices, to their placement in compact clusters translates into a reduction in overall spatial entropy.

Figure 3(a) depicts the variation of the spatial entropy for different grain sizes ($3\times$ 3 and 6×6) for a graph $G_{GAR}(8, 25, 0.25, 0.0032)$. The large rapid decrease corresponds to a rapid initial clustering. Isolated vertices are quickly moved to locations of high local density (i.e. close to the positions of other vertices with which they are highly linked) and a number of clusters are quickly established, each containing vertices of the same expected class (Figure 4(b)). However the quantity of clusters at this stage greatly exceeds k. Small clusters are unstable, since the surrounding empty grid locations lead to relatively small values of f. Hence vertices within these clusters are moved to join other clusters, resulting in fewer, larger clusters, as shown in Figure 4(c). The entropy measurement shows this phase quite clearly - while spatial entropy with grain size $3\times$ 3 stabilises, the entropy measurement for 6×6 grains continues to fall. Finally, stochastic fluctuations at the borders of clusters lead to the domination of one cluster over others corresponding this area "attracts" more and more dissimilar vertices, resulting eventually in all the clusters lying in the same part of the grid, with little or no separation between them. If α is small then the algorithm tends to produce many, small classes. The most appropriate values for a were found by experimentation to lie in the range $[0.8, 1[$.

- Number of automatons : it should be fairly small with respect to the number of vertices n. A large number results in the displacement of too many vertices at any one time which disturb local optimisations : n should exceed the number of automatons by approximately an order of magnitude.
- Grid size : the quantity of grid positions should exceed the quantity of graph vertices, n, by roughly an order of magnitude . Smaller grids inhibit class separation, since they do not have sufficient unoccupied positions, whereas larger grids, due to the increased search space involved, require more time steps to achieve classification.

In order to speed up the algorithm two slight modifications have been add to the basis version. Automaton ah such that $s_t(a_h) = (\Pi_t(a_h),\ 0)$ is directly displaced onto a vertex position chosen randomly. If $s_t(a_h) = (\Pi_t(a_h), v_i)$ and the position chosen to place v_i is already occupied, an unoccupied position on a connected cell of $\Pi_t(a_h)$ is chosen. Let us just notice that although the computational time was not here our main interest, due to the relatively simple calculations employed by the algorithm, it is low at 3 or 4 minutes on a 486 Personal Computer for a graph containing 500 vertices.

3.2 Results on Random Graphs

To evaluate the performances of the algorithm we consider a well-known class in VLSI layout of random graphs with an "expected structure" (Garbers and al. 1990). Let us denote by $G_{GAR}(k, nc, p_{int}, p_{ext})$ a graph containing k clusters of nc vertices each, where the probability of an edge vi, vj is p_{Sint} if v_i and v_j belong to the same cluster (i.e. if $i \equiv j$ (modulo nc)), and p_{ext} otherwise, all edges being chosen independently. Figure 1 shows the representation of $G_{GAR}(4, 25, 0.167, 0.0032)$ after 200000 iterations of the algorithm ; four compact and well separated clusters containing vertices 1-25, 26-50, 51-75, and 76 -100 are easily distinguishable, reflecting the expected underlying structure of the graph.

The algorithm has been successively applied to graphs $G_{GAR}(k, nc, p_{int}, p_{ext})$ containing up to 500 vertices, with a variety of class sizes from $nc = 10$ to 50, with $p_{int} = O(nc - 1/2)$. In all cases, the expected classes are reflected on the grid by easily identifiable clusters. An application of factor analysis to the same graphs with the same dissimilarity on vertices shows that, for graphs with large k, a representation on the plane R^2 does not allow all clusters to be distinguished ; as already shown by Fraysseix and al. (1993) additional dimensions are needed. A comparison on a graph of $G_{GAR}(10, 10, 0.4, 0.004)$ is presented Figure 2. For the algorithm presented here, different experiments show that for large graphs it is sufficient, for the algorithm presented here, to increase the grid size to obtain a legible representation.

3.3 Quantitative evaluation

Clustering is achieved by interactions on a local scale, hence no global criterion is required. In order to evaluate the global success of the algorithm quantitively, and to aid in explaining its operation, we introduce two measures which reflect the visual state of the grid representation as the algorithm proceeds.

adjacent to the vertex v_i , and includes v_i :

$$\rho(v_i) = \{v_j \in V;\ \{v_i,v_j\} \in E\} \cup \{v_i\}$$

We propose the use of a dissimilarity which has been applied to problems of pagination (Fraysseix and Kuntz 1992) ; it reflects properties of local density and is aimed at regrouping in the same class those vertices having a number of common neighbours and few distinct neighbours :

$$d\,(v_i,v_j) = \frac{|\rho(v_i)\Delta\rho\,(v_j)|}{|\rho(v_i)| + |\rho\,(v_j)|}$$

where Δ designates the symmetric difference. In describing the graph by its adjacency matrix, d can be recognised as the Czekanowski - Dice coefficient which was originally defined to evaluate differences between binary data in ecology (Dice 1945). Its geometric properties have been studied by several authors (e.g. Gower and Legendre 1984 ; Joly and Le CalvŸ 1994) ; d is a l_1-distance. Some heuristics for isometric representations or approximations according to classical global criteria in multidimensional scaling have been recently proposed (Hubert and Arabie 1992) but their complexities are too high for applications on large data sets. Here we relax such requirements for the representation on $(\Gamma,\ d)$. The dissimilarity d carries only local information : it takes the same value, 1, for all pairs of vertices which are non-adjacent and without common neighbours. Consequently the representation is geared to preserving on the one hand the isometry for the small values of d, and on the other just the order between the small and the large values ; the relative positioning of clusters here is arbitrary.

3 Computational Experiments

3.1 Choice of Parameters

The performances of the algorithm depend on a set of independant parameters which concern the probabilities of placement/displacement, the local density function, its distributed character with the number of automatons and the size of the grid for graph representation. Nevertheless this latter is robust in the sense here that the simulations show the stability of its results for a wide range of parameters settings. Their respective efficient values are given below.

- Probabilities of placement/displacement : The constants k_p and k_d determine the extent to which the probabilities $p_p(v_i)$ and $p_d(v_i)$ depend on the local density function f. Numerical experimentation has revealed that they affect rather the speed with which clustering is achieved rather than the quality of clustering, with the exception that large values lead to unstable clusters if class cardinalities are small (≤ 20). For the runs described below they were set within the range 0.03 to 0.1.

- Local density function : since algorithm relies on interactions on a local scale, the local area size S should be small, i.e. 3×3 or 6×6. The constant $\alpha \in [0,1]$ has, as described in section2.1, the dual role of scaling dissimilarities so that close vertices end up in the same cluster, and ensuring a good separation between clusters. If $\alpha = Maxd(v_i,v_j); v_j \in S$ then $f(v_i)$ yields relatively high values even for areas containing a mixture of vertices with both low and high dissimilarity to v_i. This leads to poor separation of the classified clusters : such areas would form in practice around a vertex with intermediate values of d, usually around the cluster boundaries. The high value of $f(v_i)$ associated with

```
1. Select $a_h$ at random from $A$.

2. Vertex displacement :
   If $s_t(a_h) = (\Pi_t(a_h),\ v_i)$ then
       - place $v_i$ in $\Pi_t(a_h)$ (i.e. $\Pi_{t+1}(v_i) = \Pi_t(a_h)$)
           with probability $p_p(v_i)$.
       - increment $list(a_h)$ by $(v_i,\ \Pi_{t+1}(v_i))$
   If $s_t(a_h) = (\Pi_t(a_h), 0)$ and $\exists v_i \in V$ s.t. $\Pi_t(v_i) = \Pi_t(a_h)$ then
       - displace $v_i$ (i.e. $s_{t+1(ah)} = (\Pi_{t+1}(a_h),\ v_i)$) with
         probability $p_d(v_i)$
       - choose $v_i^\star \in list(a_h)$ s.t $d(\Pi_t(a_h), P(v^\star)) =$
         $Min\{d(\Pi_t(a_h), P(v_i)); v_i \in list(a_h)\}$
       - set $\delta_h = 0$ and $\delta_h^\star = \delta_{h1}^\star + \delta_{h2}^\star$ (i.e
         $\delta_h^\star = \delta_1(\Pi_t(a_h),\ P(v^\star)) + \delta_2(\Pi_t(a_h),\ P(v^\star)))$

3. Automaton displacement :
   If $s_t(a_h) = (\Pi_t(a_h),\ 0)$ then
       - displace $a_h$ by $r$ grid elements in a random
         direction.
   If $s_t(a_h) = (\Pi_t(a_h),\ v_i)$ then
       if $\delta_h \le \delta_h^\star$ then                                  otherwise
               - displace $a_h$ by $r$ grid elements in a
                 random direction with probability $p_r$
       otherwise
               - displace $a_h$ by $r$ grid elements in the
                 direction of the x-axis (resp. y-axis)
                 towards $v^\star$ with probability
                   $\delta_h = \delta_h + \delta(\Pi_t(ah), \Pi_{t+1}(a_h))$
       - displace $a_h$ by $r$ grid elements in a random
       direction
```

The probability p_r just introduces a small noise to cope with certain configurations which would otherwise engender blockages, i.e. where automata prevent eachother from moving. An simple example is two automata lying on the same grid row, separated by r elements. Each is being displaced towards vertices in their respective tabu lists which also lie on the same row, but beyond the position of the other automaton. The probability p_r was set to 0.05 for all runs.

2.2 Dissimilarity on the Vertex Set

In embedding the abstract graph in a metric space E the aim is to convey explicitly through the subsequent geometric representation, the relationships of interdependence between the graph's vertices upon which an efficient partition is based. Here, these relationships are transferred into the new representation by means of a dissimilarity d, the representation in E having to preserve a certain isometry with respect to d. Generally speaking, the choice of the dissimilarity is a function of both the information available on the graph, and any constraints imposed by the desired application. The information may be of various natures : it may concern, in the case of electronic circuits, e.g. the physical properties of the circuit components or electrical network on which the graph is modelled (e.g. Lagognotte 1991), or it may concern the topological characteristics of the abstract graph (Buckley and Harary 1990). We deal with the latter case : here, the dissimilarity between any two vertices of G is calculated from the simple description of relationships of adjacence between the vertices of V. Let $r(v_i)$ denote the set of vertices of V which are

2.1 The Ant Clustering Algorithm

We present an adaptation of the principles of Lumer and Faieta's algorithm based on the behavioural model of the ant colony proposed by Deneubourg and al. A set A of finite-state automata a_h is initially layed out at random positions on Γ, as are the vertices of V (as here no ambiguity is possible we will continue to denote the embedded vertex on the grid by v_i). Let $\Pi_t(a_h)$ and $\Pi_t(v_i)$ denote respectively the positions of a_h and v_i on Γ at each descrete time step t. At each t, an automaton of A is selected at random. If a vertex lies on its position, the automaton can displace the vertex onto another position. Otherwise the automaton changes its own position in a random direction. The act of displacing the vertex onto its new position is carried out in a probabilistic manner and may take several time steps. Since a different automaton is selected at t, a number of automata and vertices may be in the process of displacement at any one time, resulting in an effective parallel operation.

The probability $p_d(v_i)$ that an automaton will displace a vertex v_i increases the more v_i is isolated, i.e. where the number of similar vertices in the immediate neighbourhood is small. By the same token the probability $p_p(v_i)$ that an automaton will place a vertex v_i onto a new position increases with the number of similar vertices in the immediate neighbourhood. These probabilities are defined by

$$p_d(v_i) = \left(\frac{k_d}{k_d + f(v_i)}\right)^2$$

and

$$p_p(v_i) = \left(\frac{f(v_i)}{k_p + f(v_i)}\right)^2$$

with k_d and k_p constants

The local density function f represents an estimation of the density of similar vertices to v_i in its neighbourhood, defined here by an area S of $\sigma \times \sigma$ elements of Γ in which v_i lies at the centre

$$f(v_i) = \begin{cases} \frac{1}{\sigma^2} \sum_{v_i;\Pi_t(v_i)\in\Sigma} \left(1 - \frac{d(v_i,v_j)}{\alpha}\right) & iff(v_i) > 0, \\ 0 & otherwise \end{cases}$$

Note that while f can never be negative, the same is not true for the expression within the sum. A highly dissimilar vertex to v_i is less desirable in Σ than an unoccupied position, hence the constant a scales the dissimilarities so that dissimilar vertices will lead to a reduction in the overall value of f. This aspect leads ultimately to visible separation of classified vertex clusters on Γ since any area containing both similar and dissimilar vertices will yield a low value of f, resulting in the displacement of any vertex within. The maximum value of f is reached if and only if all the elements of Σ are occupied with vertices v_j such that $d(v_i, v_j) = 0$, in which case $f(v_i) = 1$. The search for a new position for a displaced vertex progresses by local optimisations. Each automata ah is assigned a tabu list $list(a_h)$ containing the m pairs $(v_i,\ P(v_i))$, corresponding to the m last vertices v_i displaced by a_h and their new positions $P(v_i)$ at the instant of placement. The displacement of a new vertex v_j is carried out by a random walk (described below) with a heavy bias in the direction of the closest vertex in $list(a_h)$, in terms of dissimilarity d, to v_j.

In the following, $\delta = \delta_1 + \delta_2$ denotes the l_1-distance on the grid Γ, and δ_1 (resp. δ_2) its projection on the x-axis (resp. the y-axis). At each t, the state $s_t(a_h)$ of the automaton a_h is defined by its position and either the vertex it is currently displacing, or 0 if it is not displacing a vertex : $s_t(a_h) = (\Pi_t(a_h),\ v_i)$ or $s_t(a_h) = (\Pi_t(a_h),\ 0)$. The algorithm proceeds in discrete time steps tS, each comprised of the following 3 phases :

When solving problems of partitioning, it is usual to assign to the circuit in question a graph $G = (V, E)$ where V is the set of vertices and E the set of edges, such that each edge connects two vertices. The vertices of V symbolise the different elements (transistors, gates or more complicated subcircuits called blocks, for VLSI circuits) and the edges of E symbolise the physical connections between these elements. Both vertices and edges can be weighted ; such weights reflecting respectively for example the area taken up by a component and the multiplicity or importance of a wiring connection. Recent years have seen the appearance of circuits described by hypergraphs. Certain publications propose partitioning algorithms which are directly applicable to hypergraphs, whilst others suggest prior transformation of hypergraphs into graphs (Lengauer 1990). We restrain ourselves here to the case of graphs G for which the edges can be given positive weights, and are concerned with partitions of the set V of the graph's vertices, i.e. of the elements obtained by removal of the edges.

1.1 Graph Partitionning

1.2 Graph Clustering

1.3 Clustering in Ant Societies

The algorithm originates from the findings of entomologists who, on observing societies of ants, have remarked that larvae and food are not scattered randomly about the nest, but in fact are sorted into homogenous piles. Deneubourg and al. (1990) proposed a behavioral model where the spatial structure of the nest emerges as a result of simple, local interactions without the need for any centralised control or global representation of the environment. In this model, the environment is a two dimensional grid upon which is scattered a set of objects, each having random initial positions, and comparable with eachother by an equivalence relationship. Each ant is modelled by an automaton which is able to to move on the grid, and displace the objects according to probabilistic rules necessitating only local environmental information. The combined actions of a set of these automata lead to the grouping in the same spatial region of objects belonging to the same class of equivalence. This model has been applied with success in robotics to demonstrate the possibility of accomplishing complex tasks quickly, using several simple robots instead of a single complex one (Beckers and al. 1994). Lumer and Faieta (1994) have recently extended the model to work with objects which are comparable according to a measure of dissimilarity; their algorithm forms one or more spatial groups such that similar objects belong to the same group, and dissimilar ones belong to different groups, each group being spatially distant from one another. In contrast to multidimensional scaling methods, this is a distributed algorithm, in the sense that no global optimisation criterion is calculated ; only local optimisations (by the automata) are carried out. Analysis of computer simulations shows that the representation achieved preserves the order between small and large values of dissimilarity. This approach was originally tested using Euclidean distances. We extend it here to the representation of a graph with a view to its partitioning, by defining a dissimilarity on a set of the graph's vertices.

2 A New Graph Clustering Algorithm

We take a dissimilarity d as being defined on the set V of graph vertices. The grid on which the graph is represented is denoted by Γ.

An Ant Clustering Algorithm Applied to Partitionning in VLSI Technology

Pascale Kuntz[1], Paul Layzell[2]
[1]Ecole Nationale Supérieure des Télécommunications de Bretagne
BP 832, 29285 Brest Cedex, France
Fax: (33) 98 00 10 30
Email: Pascale.Kuntz@enst-bretagne.fr
[2]University of Sussex - Falmer - Brighton - England
Email: paulla@cogs.susx.ac.uk

Abstract

For papers submitted to ECAL97 an abstract of about 100 words is required here.

1 Introduction

The size of networks of various relationships met in a wide variety of domains from e.g. neuroscience to telecommunications is continually growing and, in order to combat frequently inextricable calculations necessary for their analysis, different techniques increasingly require a preliminary stage of segmentation. The domain which has most probably produced the largest number of publications on this problem is that of Computer Aided Design (CAD). VLSI (Very Large Scale Integration) circuits, who have known a developmental explosion during the 1980s, are very often too complex to be designed or analysed on a global basis. Hence partitioning lies at the root of numerous CAD problems, notably (e.g. Davis-Moradkan 1990 ; Lengaueur 1990 ; Sait and Youssef 1995) :

- pagination, in which networks are divided up so that they can be represented legibly using automatic line-tracing software over several standard sized pages ;
- logical tests carried out during the manufacture of integrated circuits to ensure reliability. The classical method of testing a small circuit composed of logic gates is to apply successively each binary configuration possible at its inputs. The subsequent output states then permit verification that the gates are functioning correctly. For large circuits, such an exhaustive approach is impractical, and an alternative method is to decompose them into testable subcircuits of smaller size ;
- automatic circuit placement and routing, which consists of arranging the circuit components on a surface, and linking them using conductive strips, or routes. This fundamental phase in circuit design is a complex process which must cope with different objective functions (minimizing area taken by components, minimizing total route lenght, ...) and numerous various constraints (avoiding overlap of layout cells and route congestion, satisfying topological requirements imposed by the technology, ...). Due to this complexity it is generally carried out using "top-down" hierarchical approaches and, in practice, the most current algorithms proceed with successive divisions of the circuit.