Complex Systems **3** (1989) 531–575

Neural Networks and Graph K-Partitioning

Laurent Hérault
Jean-Jacques Niez
CEA-IRDI Division LETI/D. SYS CENG,
Avenue des Martyrs, 85X 38041 Grenoble Cedex, France

Abstract. With the emergence of neural network architectures, combinatorial optimization problems and NP-complete problems may be tackled with a new attention combining biology, physics and data processing. This paper deals with one of these problems: the graph K-partitioning. After a brief critical review of the conventional methods, we show how a particular vectorial encoding associated with this problem produces original neural network methods. Through different graph families, a comparative analysis of our approaches with one of the best conventional algorithms is developed.

1. Introduction

The graph partitioning, when it is subject to some particular constraints, is a NP-complete problem [5] having a lot of potential applications. One of them concerns the optimal assignment of distributed modules to several processors in order to minimize the cost of running a program. This cost may be money, time or some other measures of resource usage. Another application is the layout of micro-electronic systems: one wants to assign small circuits to packages (chips) of specified sizes in order to minimize one measure of interconnection between them.

1.1 Graph partitioning and computer vision as an example

This problem appears in the field of computer vision where we expect a lot of applications. The first of them concerns the perceptive grouping. In fact, salient features in an image may be described as image entities represented by the vertices of a graph. Topological relationships exist between them, the latter being represented by weighted edges.

The second application, here considered for information only and using nonhomogeneous graphs, concerns the stereo-correspondence. One wants to match two images of the same scene from different viewing positions in order to extract 3D-informations of the scene. The best methods need graphs to

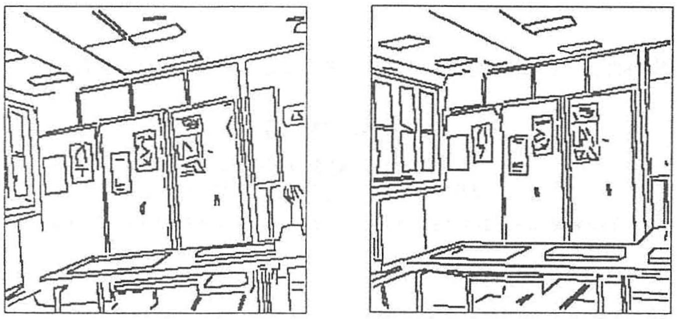

Figure 1: Example of segment images extracted from two views of the same scene. The left image has 323 segments and the right one has 314 segments.

reduce the combinatory and produce valuable results as well [1,9]. Here we use the method developed by Horaud and Skordas [9]. Segments are first extracted from both left and right images (see figure 1). Each segment is characterized by its position, orientation and some topological relationships with its nearby segments. So, monocular descriptions of each image are represented as graphs (see figure 2). Each vertex represents a segment and a weighted edge between two vertices is associated to a topological relationship between two segments in the image (left of, right of, colinear with, same junction as). Those two graphs are generally nonhomogeneous and have to be matched. But they are so complex that it is necessary to partition them into subgraphs in order to make a parallel treatment. The cost of the partition is measured by the total sum of all edge weights between vertices of distinct subsets in the partition. So far, the authors have used an arbitrary way of partitioning: they cut images in slightly overlapping windows (see figure 3). In their case, the subset number is a power of 4. One notices that the partitioning does not take into account the nonhomogeneity of the total graph. Consequently, subsets may be largely unbalanced and the interconnection cost may be very high. In fact, salient structures in the image corresponding to high local topological relationships may be broken (see figure 4). Therefore it is necessary to impose some constraints on the partition. Every subgraph of an image is matched with the entire graph of the other image. So, a first constraint must be imposed: the interconnection cost between the subgraphs must be as small as possible. On the other hand, in order to optimize the running of the parallel matching, we have to impose the following second constraint: the subsets must have specified sizes in order to have a good load balancing between processors.

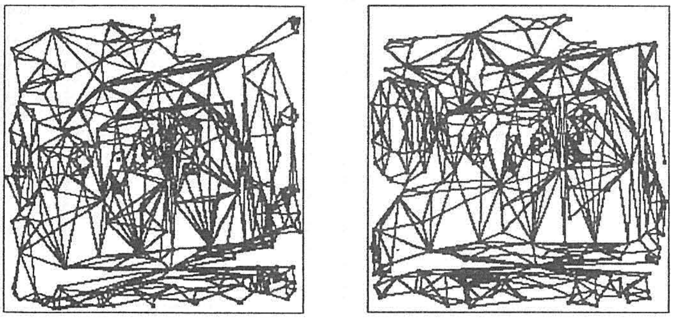

Figure 2: Monocular descriptions associated with the figure 1 images. They correspond to nonhomogeneous graphs in which every vertex represents a segment and an edge between two vertices represents a topological relationship. The left monocular description has 323 vertices and 910 edges. The right one has 314 vertices and 874 edges.

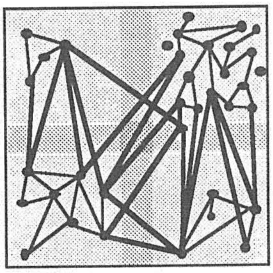

Figure 3: Example of an arbitrary partition by slightly overlapping windows.

1.2 Theoretical formulation of the graph K-partitioning problem

Given an undirected graph $G = (V, E)$ of N vertices and M positively weighted edges, one wants to partition this graph into K distinct sets of specified sizes $N_1, \ldots, N_K$ in order to minimize the total weight of edges connecting vertices in distinct subsets. Let $A = (a_{ij})$ be its weighted adjacency matrix. One defines the density d of a graph as the ratio between M and the number of edges in a complete graph of N vertices. So, the average

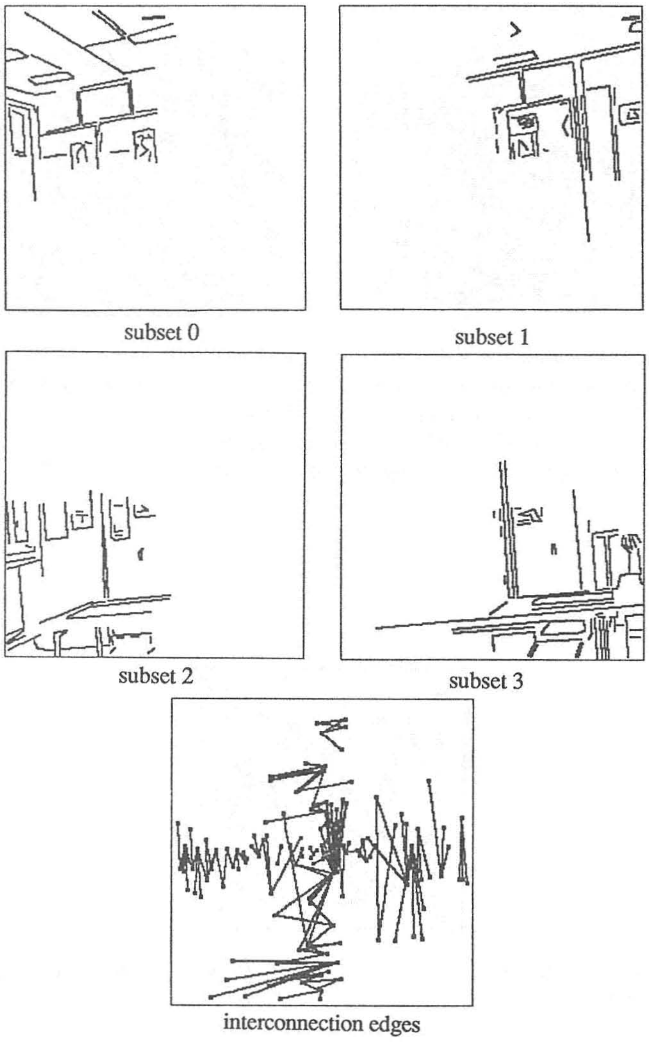

Figure 4: Example of an arbitrary partition of the left monocular description. The distribution of vertices is the following: 74 vertices in subset 0, 75 vertices in subset 1, 78 vertices in subset 2 and 96 vertices in subset 3. One notices that the subsets are quite unbalanced and that the interconnection cost is high.

degree of a vertex, i.e. the average number of vertices connected to a vertex is $N.d$. The standard deviation of the degrees appraises the graph homogeneity. Thus we consider two graph families: the family of small standard deviation graphs, named homogeneous graphs, and the family of nonhomogeneous graphs. The more nonhomogeneous the graph, the more necessary and nonobvious the partitioning. Henceforward, we will use these different graphs to test the methods here proposed. In all cases, one can prove that the interconnection cost of a perfectly balanced partition ($N_1 = \ldots = N_K$) is a function of K and d which is bounded by $C_{\min}$ and $C_{\max}$ given by (see appendix A):

$$\begin{aligned} C_{\min}(K) &= 0 \text{ if } K \leq \frac{N}{(N-1).d+1} \\ &= \frac{N^2.d}{2}.\left[1 - \frac{1}{K}.\left(\frac{d-1}{N.d}.K + \frac{1}{d}\right)\right] \text{ otherwise} \end{aligned} \tag{1.1}$$

$$C_{\sup}(K) = \frac{N^2.d}{2}.\left(1 - \frac{1}{K}\right), \forall K \leq N. \tag{1.2}$$

The function $C_{\min}$ is obtained by supposing that every subgraph of the partition is complete (density = 1). The function $C_{\sup}$ is obtained by supposing that the internal density of every subgraph is equal to the graph density. An example is shown in figure 5. In reality, if the graph is homogeneous and N much greater than K, we can estimate that the internal density of each subgraph is about $K.d$ (see appendix A).

Among the homogeneous graphs, we name *regular with torus pattern* the most homogeneous ones. The most famous examples of such graphs are the rectangular grid (see figure 6) and the hexagonal grid. Among the nonhomogeneous graphs, one finds the figure 2 monocular descriptions.

Two novelties are presented in this paper. The first is the formulation of an extension of neural methods (simulated annealing, Hopfield neural

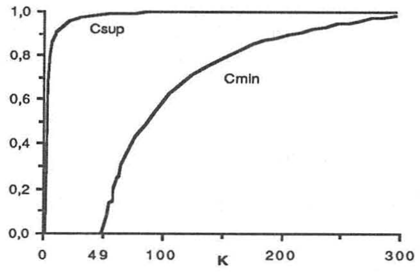

Figure 5: Curves bounding the interconnection cost, divided by M, of the left graph partition shown in figure 2.

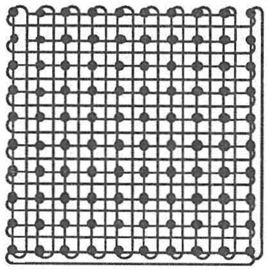

Figure 6: Example of an homogeneous graph: rectangular grid with torus pattern.

network, mean field theory, mean field annealing) to the manipulation of vectorial entities used as optimization variables. The second concerns the application to the graph K-partitioning problem. This paper is organized as follows. In section 2, we show that an exhaustive production of all the solutions is impossible. In section 3, we briefly review previous conventional approaches of the problem. In section 4, we develop neural methods using a new vectorial encoding.

2. Exhaustive production of solutions

Let us suppose that one wants to explore exhaustively the space of the possible distributions of $N.K$ objects into K subsets of size N. The total number of feasible partitions is

$$\frac{1}{K!}\cdot\binom{N.K}{N}\cdot\binom{N.K-N}{N}\cdots\binom{2.N}{N}\cdot\binom{N}{N}=\frac{(N.K)!}{K!\,.\,(N!)^{K}}\quad(2.1)$$

Typically, with $N.K = 250$ and $K = 10$, the number of configurations to study is greater than 10^{234}. Such an exploration would need thousands of years of CPU time of the most powerful computers. So there is no question of developing exhaustive methods to solve such a problem: we have to develop some heuristics.

3. Previous conventional approaches

Three classical method families emerge from the sixties.

3.1 Linear programing

In the past, our problem has been considered as a linear programing problem. The first who has described the problem in such terms is Lawler [13]. In this framework, a lot of methods have been developed. Among them, Donath [4] proposed to use the eigenvectors of a modified adjacency matrix; IBM researchers [16] used a Cholesky factorization of a modified adjacency matrix to iteratively improve a partition which is the solution of a linear system of $N.K$ variables. Lukes [14,15] used a dynamic programing procedure to generate a good partition. All these methods are not adapted to problems involving large size graphs: they are inextricable. Moreover, their parallel implementation seems to be very difficult.

3.2 Use of the Ford-Fulkerson maxflow–mincut theorem

Another approach to solve only the bipartitioning problem is to consider the graph as a network of pipes conveying some commodity from a source vertex to a sink vertex. The edge weights represent the capacities of the pipes. Stone [20] and Bokhari [2] used the maxflow–mincut theorem to solve the problem of the optimal assignment of modules on two processors. But this approach does not allow one to impose the sizes of the two subsets. Therefore, the problem is not NP-complete. Rao [18] studied this problem when the memory size of each processor is limited. More generally, it seems very difficult to extend successfully those methods to the K-partitioning problem.

3.3 Iterative improvements

Burnstein [3] has made a review of iterative improvement heuristics and considered two heuristic families to solve the bipartitioning problem: methods of constructing a good initial partition and methods of improving an initial partition. Very few of them produce good results because most tend to converge on the first found local minimum. Nevertheless one of them, proposed by Kernighan [10,11], rapidly produces a very good bipartition. The idea is the following: given an initial graph bipartition which is perfectly balanced, the optimal bipartition may be obtained by interchanging a vertex group of one subset of the bipartition with a vertex group of the other subset. In order to approximate those vertex groups, one executes a sequence of vertex permutations from one subset to the other so that globally the interconnection cost decreases. Thus the algorithm allows a temporary increase of the interconnection cost. By reason of that, this method keeps from being trapped at the first local minimum: one is able to leave shallow valleys of the solution landscape. The more homogeneous the graph, the better the final partition because the depth valley disparity is small. The major drawback of this approach is its sensibility to the initial partition quality.

We have extended this approach to the K-partitioning problem with $N_1 = \ldots = N_K = N/K$ by a dichotomic recursive procedure illustrated in figure 7 (case of the 7-partitioning).

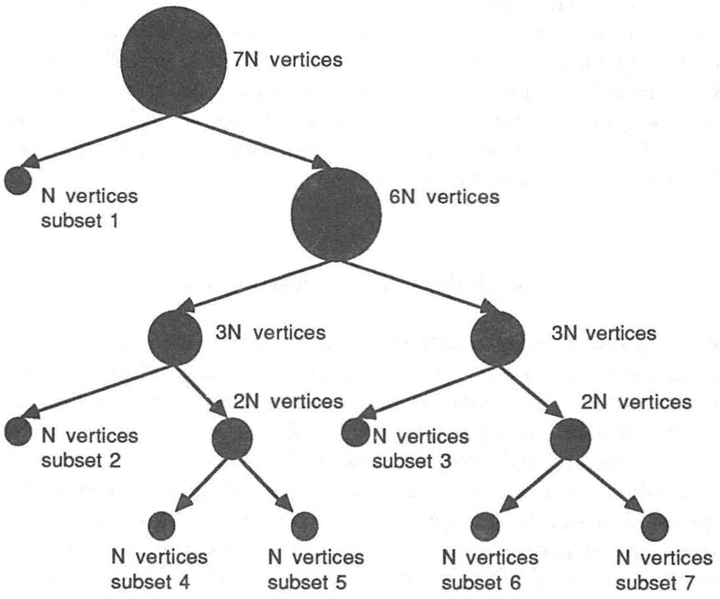

Figure 7: Dichotomic recursive procedure for the 7-partitioning using the generalized Kernighan method.

Experimental results are presented for the 5-partitioning of the following graphs:

homogeneous graph: regular hexagonal network of 324 vertices (see figure 8),

nonhomogeneous graph: left monocular description of figure 2 (see figure 9).

Experimentally, this heuristic gives insufficient results when the desired subset number is greater than 4. In fact there is a contradiction in the dichotomic way of partitioning: first, a bipartitioning procedure tries to maximize the internal connection cost in a subset (i.e. minimize an interconnection cost), and then a new bipartitioning procedure imposes to minimize a connection cost in this subset. So, the greater K, the worse the result.

4. Neural approaches

The following approaches are the result of a conjunction between biology, physics and data processing. Solving an optimization problem subject to

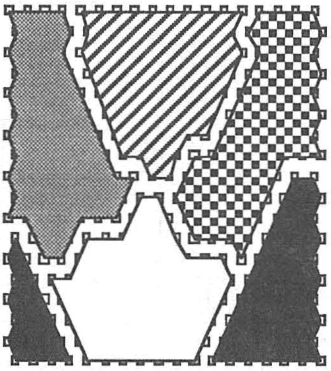

Figure 8: Balanced 5-partitioning of a regular hexagonal network of 324 vertices and 901 edges provided by the generalized Kernighan method. The interconnection cost is 107 (edges are not shown).

constraints such as the graph K-partitioning one is equivalent to minimizing a global quadratic energy which describes the partition state. Here we present some original neural methods to minimize such an energy. Those methods differ from previous approaches (see section 3) by the following characteristics:

their ability to relax constraints (this is desirable for the partitioning of nonhomogeneous graphs),

they can easily be implemented on massively parallel architectures such as neural networks; the initial vertex state does not noticeably influence the final partition quality,

they give good results whatever the number of desired subset,

they can be easily extended to solve a lot of other combinatorial optimization problems.

A network of formal neurons is a set of highly interconnected processing elements which imitate biological neurons. A formal neuron is defined by

an internal state (identical to the output),

connections with some other neurons or with the environment,

a nonlinear transition function which allows to calculate the internal state as a function of the signals received on its synaptic connections.

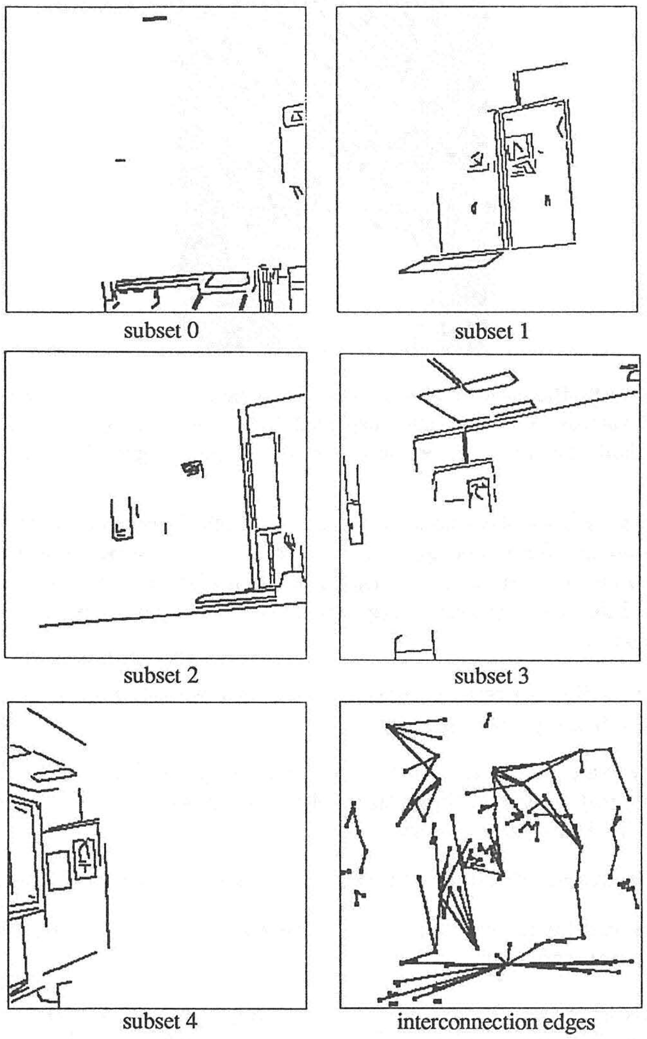

Figure 9: 5-partitioning of the left nonhomogeneous graph of the figure 2 provided by the generalized Kernighan method. The subsets are perfectly balanced. The interconnection cost is high: 125.

A synapse between two neurons is represented by a weighted connection between the output of a neuron and one input of the other. Despite the extreme simplicity of this model, collective computations with formals neurons are particularly well suited to solve combinatorial optimization problems [21,22].

We distinguish two neural network families: networks with binary neurons and networks with analog neurons. The following algorithms take into account this characteristic feature of the neurons.

4.1 Transcription of the optimization problem in terms of energy

We associate to every vertex a vector which defines its localization in the partition:

$$\vec{V} = [V^1 \dots V^K]^t \tag{4.1}$$

where $V^k = 1$ if the vertex is the subset k and $V^k = -1$ otherwise.

Let us calculate the partition interconnection cost. First, we notice

$$\begin{aligned} a_{ij}.\left(\frac{V_i^k - V_j^k}{2}\right)^2 &= a_{ij} \text{ if one and only one of the vertices } i \\ &\quad\ \text{and } j \text{ is in the subset } k, \\ &= 0 \text{ otherwise.} \end{aligned} \tag{4.2}$$

Therefore, the interconnection cost between the subset k and the other subsets is

$$\frac{1}{2}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}.\left(\frac{V_i^k - V_j^k}{2}\right)^2. \tag{4.3}$$

The total interconnection cost between all the subsets, which we name the interconnection energy, can be written

$$E_{\text{interconnection}} = \frac{1}{2}.\sum_{k=1}^{K}\left\{\frac{1}{2}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}.\left(\frac{V_i^k - V_j^k}{2}\right)^2\right\}. \tag{4.4}$$

After some algebra, we get

$$E_{\text{interconnection}} = -\frac{1}{8}.\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}.V_i^k.V_j^k + \frac{K}{8}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}, \tag{4.5}$$

i.e.

$$E_{\text{interconnection}} = -\frac{1}{8}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}.\vec{V}_i.\vec{V}_j + \frac{K}{8}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}. \tag{4.6}$$

Now let us define an energy function which expresses the imbalance of the partition. First we notice that if the partition is perfectly balanced, then in a subset k, $N_k V_i^k$ equal $+1$ and $N - N_k$ equal -1. Therefore:

$$\forall k \in \langle 1, K\rangle, \sum_{i=1}^{N} V_i^k = 2.N_k - N. \tag{4.7}$$

An imbalance measure in the subset k is defined by

$$D_k = \left(2.N_k - N - \sum_{i=1}^{N} V_i^k\right)^2 . \tag{4.8}$$

The total partition imbalance, which we name imbalance energy, is measured by

$$E_{\text{imbalance}} = \sum_{k=1}^{K} D_k. \tag{4.9}$$

We notice

$$\forall i \in \langle 1, N\rangle, \sum_{k=1}^{K} V_i^k = 2 - K. \tag{4.10}$$

Thus, after some algebra, one finds that the linear term of D_k is a constant for all k and it leads:

$$\begin{aligned} E_{\text{imbalance}} &= \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} V_i^k.V_j^k - 4.\sum_{k=1}^{K}\sum_{i=1}^{N} N_k.V_i^k - K.N^2 \\ &\quad + 4.\sum_{k=1}^{K} N_k^2. \end{aligned} \tag{4.11}$$

To find a good solution to this optimization problem, we associate to it an energy function E. The minimization of E must ensure a respect of the constraints and the minimization of the total interconnection cost. We define E as

$$E = E_{\text{interconnection}} + \lambda/8.E_{\text{imbalance}} \tag{4.12}$$

where λ is a parameter which allows to balance the constraints. After simplifications, we get

$$\begin{aligned} E &= \frac{1}{8}.\left\{\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda - a_{ij}).V_i^k.V_j^k - 4.\lambda.\sum_{k=1}^{K}\sum_{i=1}^{N} N_k.V_i^k\right\} \\ &\quad + \frac{1}{8}\left\{K.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij} - \lambda.K.N^2 + 4.\lambda.\sum_{k=1}^{K} N_k^2\right\}. \end{aligned} \tag{4.13}$$

It is clear that it is not necessary to keep the constants and the multiplicative factors in this energy. Then, we minimize the following quadratic energy:

$$E \approx \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda - a_{ij}).V_i^k.V_j^k - 4.\lambda.\sum_{k=1}^{K}\sum_{i=1}^{N} N_k.V_i^k. \tag{4.14}$$

In order to statistically give the same importance to the balance constraint and to the interconnection cost minimization constraint, one can estimate the value of the parameter λ (using a similar kind of approach as Kirkpatrick [12]):

$$\lambda \approx \alpha.\frac{1}{N^2}.\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij} \tag{4.15}$$

where α is an adjustable parameter always around 1.

We empirically notice that the partitioning of homogeneous graphs with a parameter α close to 1 provides a partition with an excellent balance. This can be explained by the fact that the energy landscape is quite smooth: the valley depth disparity is small. Therefore, the optimization algorithm easily moves from one energetic valley to another one until the obtention of a well-balanced partition. One the contrary, the partitioning of nonhomogeneous graphs needs a balance parameter greater than 1 because the valley depth disparity grows with the nonhomogeneity of the graph. Typically, good results are obtained with α close to 2.

In the following, experimental results will be given by supposing that the subsets have the same size ($N_1 = \ldots = N_K$).

4.2 Partition energy minimization by a network with binary neurons

In this neural network family, the internal state (i.e. the output) of each neuron is binary: the nonlinear transition function associated to a neuron is a Heavyside type function.

4.2.1 Hopfield network with binary neurons

The interconnection network of the Hopfield model is complete (see figure 10). The synaptic connection between the neuron i and the neuron j is weighted by T_{ij} which is positive (excitatory synapse) or negative (inhibitory synapse). The neuron output is a function of its inputs:

$$V_i = f\left(\sum_{j=1}^{N} T_{ij}.V_j + I_i\right). \tag{4.16}$$

The transition function f is defined by the following:

$$\begin{array}{ll} \text{if } x < 0 \text{ then} & f(x) = -1, \\ \text{otherwise} & f(x) = 1. \end{array} \tag{4.17}$$

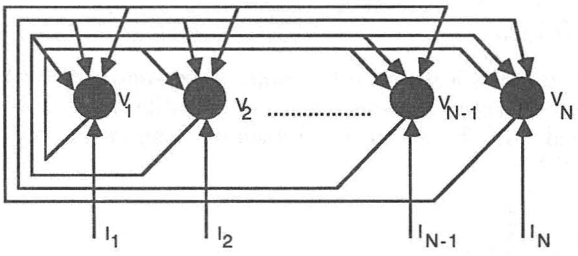

Figure 10: Hopfield neural network. The interconnection graph is complete.

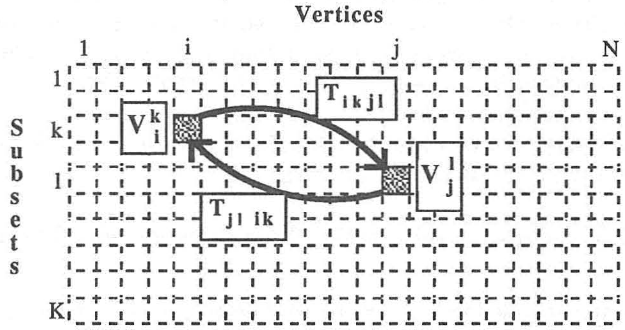

Figure 11: Matrix structure of an Hopfield network adapted to the graph K-partitioning problem.

Hopfield has shown [8] that in the case of an asynchronous dynamics, a symmetrical matrix T with 0 diagonal elements drives the system to stable states in which the outputs of all neurons are either $+1$ or -1. These stable states of the network correspond to the local minima of the quantity, which we call the energy of the system:

$$E = -\frac{1}{2}.\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}.V_i.V_j - \sum_{i=1}^{N} I_i.V_i \tag{4.18}$$

where V_i is the output of the ith neuron and I_i is the externally supplied input to the ith neuron.

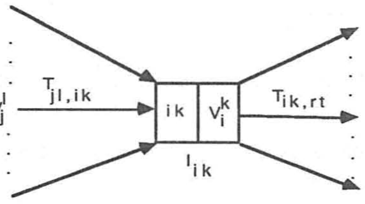

Figure 12: Details of a formal neuron of the figure 13.

Let us associate to our optimization problem an Hopfield network having a matrix organization of $N.K$ neurons in which the output of the (i,k)th neuron expresses the V_i^k value and is either $+1$ or -1 (see figure 11).

This network is a neural transcription of the vectorial representation previously defined. To each neuron is associated a processing element with several inputs and one output connected to the other neurons (see figure 12). A coefficient of T represents the synaptic weight between two neurons. The output (internal state) of the (i,k)th neuron is

$$V_i^k = f\left(\sum_{l=1}^{K}\sum_{j=1}^{N} T_{ik,jl}.V_j^l + I_{ik}\right). \tag{4.19}$$

We must add to the energy function associated to our optimization problem (equation 4.14) an energy term which takes into account the structural organization of the network (see figure 11). Therefore, we have to minimize the energy:

$$\begin{aligned} E &= \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda - a_{ij}).V_i^k.V_j^k - 4.\lambda.\sum_{k=1}^{K}\sum_{i=1}^{N} N_k.V_i^k \\ &\quad + B.\sum_{i=1}^{N}\left(2-K-\sum_{k=1}^{K}V_i^k\right)^2. \end{aligned} \tag{4.20}$$

where B is a positive adjustable parameter.

The term relative to B is an energy term which is minimum when only one component V_i^k of each neuron vector $\vec{V_i}$ characterizing a graph vertex is equal to $+1$. We get

$$\begin{aligned} E &= \sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda - a_{ij}).\delta_{kl}.V_i^k.V_j^k - 4.\lambda.\sum_{k=1}^{K}\sum_{i=1}^{N} N_k.V_i^k \\ &\quad +B.\sum_{i=1}^{N}\sum_{j=1}^{N}\left(2-K-\sum_{k=1}^{K}V_i^k\right).\left(2-K-\sum_{l=1}^{K}V_j^l\right).\delta_{ij} \end{aligned} \tag{4.21}$$

After some algebra, we find

$$\begin{aligned} E &= -\frac{1}{2}.\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}[-2.(\lambda - a_{ij}).\delta_{kl} - 2.B.\delta_{ij}].V_i^k.V_j^l \\ &\quad -\sum_{k=1}^{K}\sum_{i=1}^{N}[2.B.(2-K) + 4.\lambda.N_k].V_i^k + B.N.(2-K)^2. \end{aligned} \tag{4.22}$$

Once again, it is not necessary to keep the constant term. Our energy can be written as an Hopfield energy:

$$E = \frac{-1}{2}.\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ik,jl}.V_i^k.V_j^l - \sum_{k=1}^{K}\sum_{i=1}^{N} I_{ik}.V_i^k \tag{4.23}$$

where

$$\forall(i,j) \in \langle 1,N\rangle^2, \forall(k,l) \in \langle 1,K\rangle^2,$$

$$T_{ik,jl} = -2.(\lambda - a_{ij}).\delta_{kl} - 2.B.\delta_{ij}, \tag{4.24}$$

$$I_{ik} = 2.B.(2-K) + 4.\lambda.N_k. \tag{4.25}$$

The excitatory part of the synaptic weight (positive term) tends to put the highly interconnected vertices in the same subset but the inhibitory part (negative terms) imposes the balance and the disjunction constraints on the subsets. One notices that the matrix T of synaptic weights is symmetrical:

$$\forall(B,\lambda) \in \mathcal{R}^{+^2}, T_{ik,jl} = T_{jl,ik}. \tag{4.26}$$

Only the two parameters λ and B are necessary to calculate the matrix T. A value of B near to λ seems to be a good one because the parameters then globally balance the constraints of the problem. Then λ is the only parameter to be determined and is easily approximated (equation 4.15). Empirically, such a network converges to the nearest local minimum of the configuration hypercube.

The running on a conventional sequential architecture would be very expensive in terms of CPU time. So we have not experimentally validated this method.

4.2.2 Simulated annealing

A good way to find low energy states of a complex physical system such as a solid is to heat the system up to some high temperature, then cool it slowly. This process, called annealing, forces the system evolution into regions of low energy, while not getting trapped in higher-lying local minima. Geman [6] has shown that with an infinite initial temperature and by using an exponential law for the temperature decreasing, an absolute energy minimum is reached in an exponential number of iterations. For example, at the kth temperature step, the temperature satisfies the bound:

$$T(k) \geq \frac{c}{\log(1+k)} \tag{4.27}$$

where c is a constant independent of k. The idea of the simulated annealing is to express those concepts in terms of an algorithm. So, we identify the energy function of the system to be optimized with the energy of a physical system.

Let us consider:

a list of feasible elementary transformations which determine the energy landscape of our problem (the topology),

an initial configuration of the system,

a law of the temperature decreasing.

The smaller the temperature, the more rigid the system (there is a small number of elementary transformations which are operated) and the more deterministic the system evolution. Kirkpatrick [12] has developed this approach in the case of the graph bipartitioning problem. We extend this method to the K-partitioning problem by using the previously defined vectors.

Given the global energy to be minimized (equation 4.14), let us calculate the energy variation associated to an elementary transformation. We define an elementary transformation as the move of a vertex i from a subset k to a subset ℓ. The total number of possible elementary transformations is $N.(K-1)$. It leads:

$$\Delta E_i^{k\to\ell} = E(V_i^k = -1, V_i^l = 1) - E(V_i^k = 1, V_i^l = -1). \tag{4.28}$$

After some algebra, we obtain

$$\Delta E_i^{k\to\ell} = 4.\sum_{j=1, j\neq i}^{N} (\lambda - a_{ij}).(V_j^l - V_j^k) + 8.\lambda.(N_k - N_l). \tag{4.29}$$

New states of the system are generated by applying a set of elementary transformations to the system. Each elementary transformation is accepted or rejected using the following criterium:

if $\Delta E_i^{k\to\ell} \leq 0$, then accept the move,
otherwise accept the move with the probability

$$P(\Delta E_i^{k\to\ell}, T) = \exp\left(-\frac{\Delta E_i^{k\to\ell}}{T}\right), \tag{4.30}$$

where T is the temperature parameter.

Some curves representing $P(x,T)$ are shown in figure 13. One verifies that the acceptance probability decreases as the temperature.

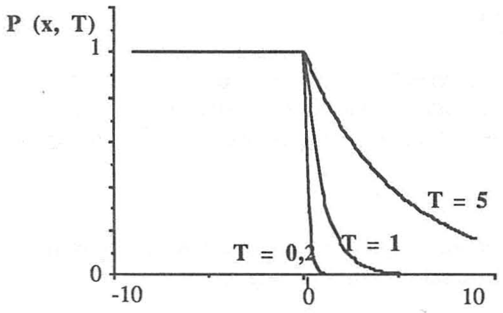

Figure 13: Simulated annealing: acceptance probabilities of an elementary transformation as a function of the associated energy variation and of the temperature T.

The principle of the simulated annealing algorithm is the following: the system is put in a high temperature environment. At this temperature is applied a sufficiently long sequence of random elementary transformations (Markov chain) to reach the equilibrium at this temperature. Then, the ambient temperature is slightly decreased and a new sequence of random moves is applied. So, the system converges slowly to a minimal energy state. This process is iterated until the system is frozen, in other words when there are not enough global significant energy improvements. By analogy to spin glass physics, one takes as a good initial temperature:

$$T_0 = (N.d)^2 \tag{4.31}$$

where d is the graph density. Here, we notice that the Markov chain length of elementary transformations which is necessary to obtain the equilibrium at a fixed temperature closely depends on the graph homogeneity. The more homogeneous the graph, the smaller the necessary length of the Markov chains because the slopes of the relevant energetic valleys are then more abrupt. This will be visualized in section 4.3.2. The simulated annealing algorithm can be found in appendix B. Experimental results are given for the 5-partitioning of the following graphs:

homogeneous graph and $\alpha = 1$: regular hexagonal network (see figure 14). Figure 15 shows the evolutions of the energies (interconnection energy, imbalance energy and total energy) as a function of the number of temperature steps. In figure 16 one shows the corresponding evolution of the temperature and of the number of accepted elementary transformations. One can see that in average those energies decrease with the temperature. Those curves are highly nonlinear: the system suddenly freezes in a certain range of temperatures.

nonhomogeneous graph and $\alpha = 1$: left monocular description of figure 2 (see figures 17, 18, and 19). One notices that $\alpha = 1$ produces a bad imbalance energy: α must be greater.

nonhomogeneous graph and $\alpha = 2$: left monocular description of figure 2 (see figures 20, 21, and 22). The 5-partition and the energies produced with such a value of α are excellent.

Those results are to be compared with those provided previously by the generalized Kernighan method (see section 3.3): the simulated annealing algorithm experimentally improves by about 20% the interconnection cost and thus the global energy of the system.

The main drawback of this method is that the running time necessary to converge on a conventional sequential computer is very high (about 3 CPU hours on a SUN 4-260 for the graphs of figures 14, 17, and 20). Nevertheless, we notice that the time complexity linearly increases as the problem size; in fact, it is imposed by the Markov chain length which is in $o(N.K)$. In order to speed up the partitioning, we will use another approach, the so called mean field annealing algorithm (see section 4.3.3). Contrary to the stochastic and sequential nature of the simulated annealing, the system evolution here

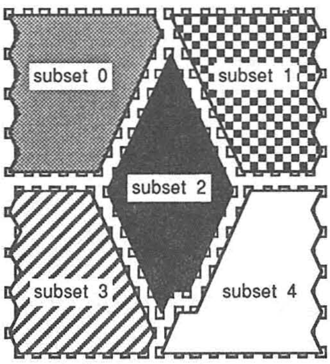

Figure 14: Balanced 5-partitioning with $\alpha = 1$ of a regular hexagonal network of 324 vertices and 901 edges provided by the simulated annealing algorithm. The interconnection cost is 85 (edges are not visualized). The imbalance energy is 99,20. The total energy is 85,21. The initial temperature is 40 and the final temperature is 0,378. The number of temperature steps is 109. The distribution of vertices is the following: 69 vertices in subset 0, 65 vertices in subset 1, 62 vertices in subset 2, 64 vertices in subset 3 and subset 4.

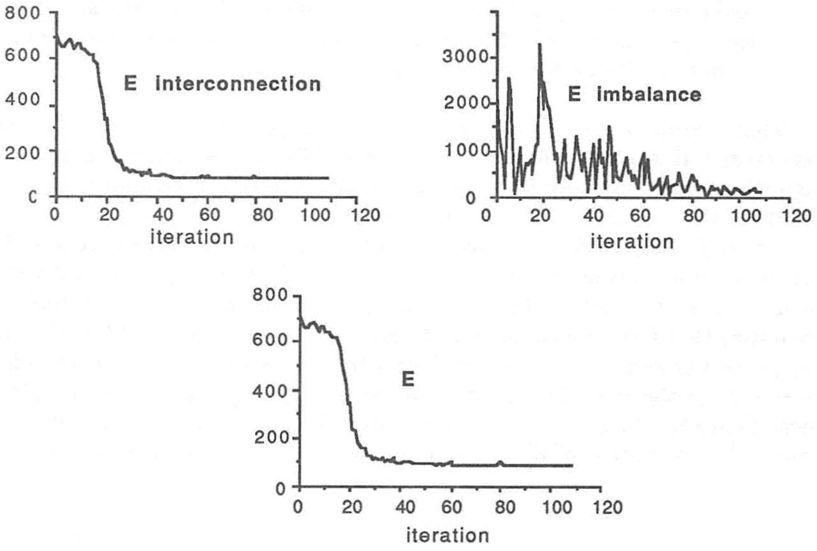

Figure 15: Evolution of the system energies as a function of the temperature step number.

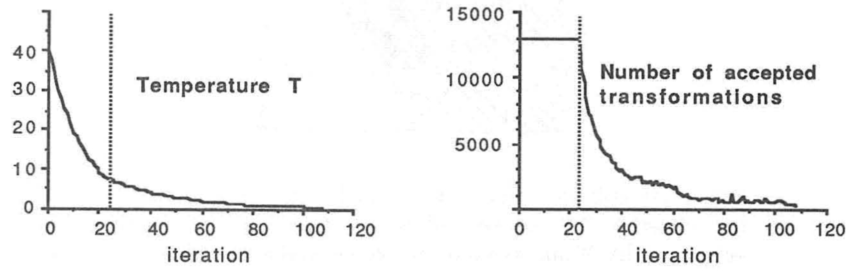

Figure 16: Evolution of the temperature and of the number of accepted transformations as a function of the temperature step number.

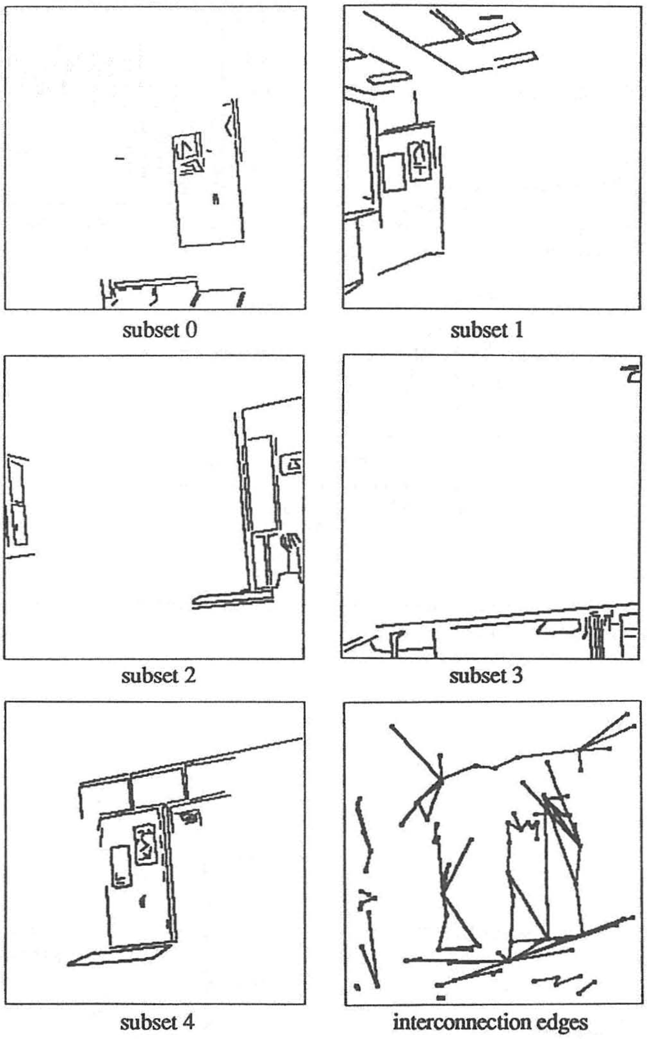

Figure 17: Balanced 5-partitioning with $\alpha = 1$ of the left nonhomogeneous graph of the figure 2 provided by the simulated annealing algorithm. The interconnection cost is 85, the imbalance energy 3572, 8, and the total energy 92, 81. The initial temperature is 40 and the final temperature is 0, 714. The number of temperature steps is 88. The distribution of vertices is the following: 53 vertices in subset 0, 69 vertices in subset 1, 67 vertices in subset 2, 48 vertices in subset 3 and 86 vertices in subset 4.

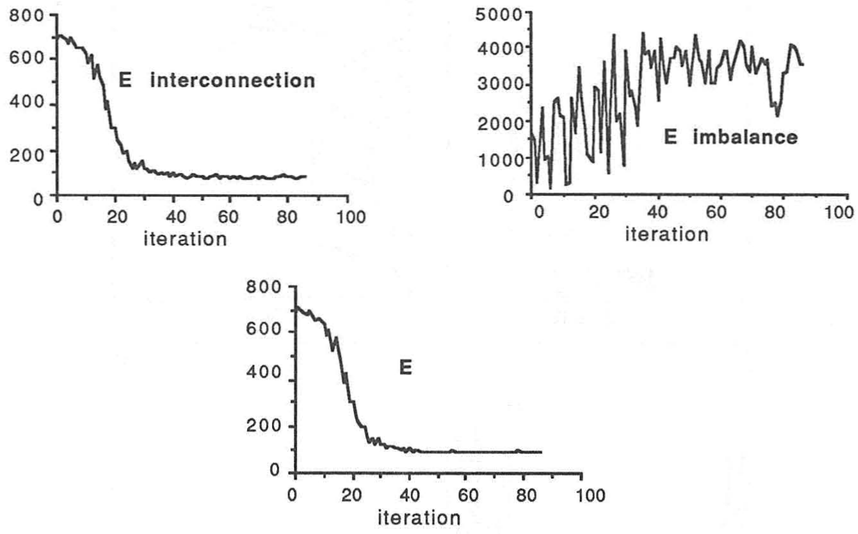

Figure 18: Evolution of the system energies as a function of the temperature step number.

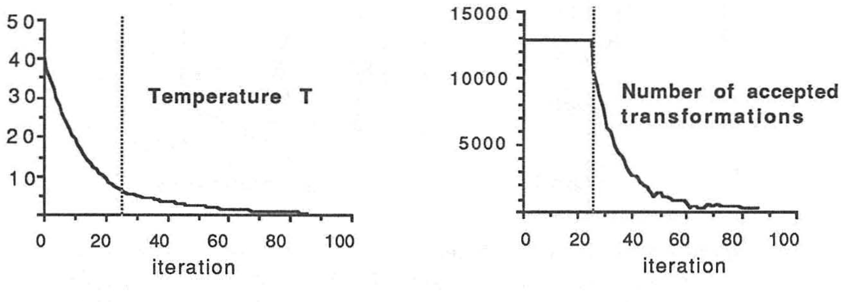

Figure 19: Evolution of the temperature and of the number of accepted transformations as a function of the temperature step number.

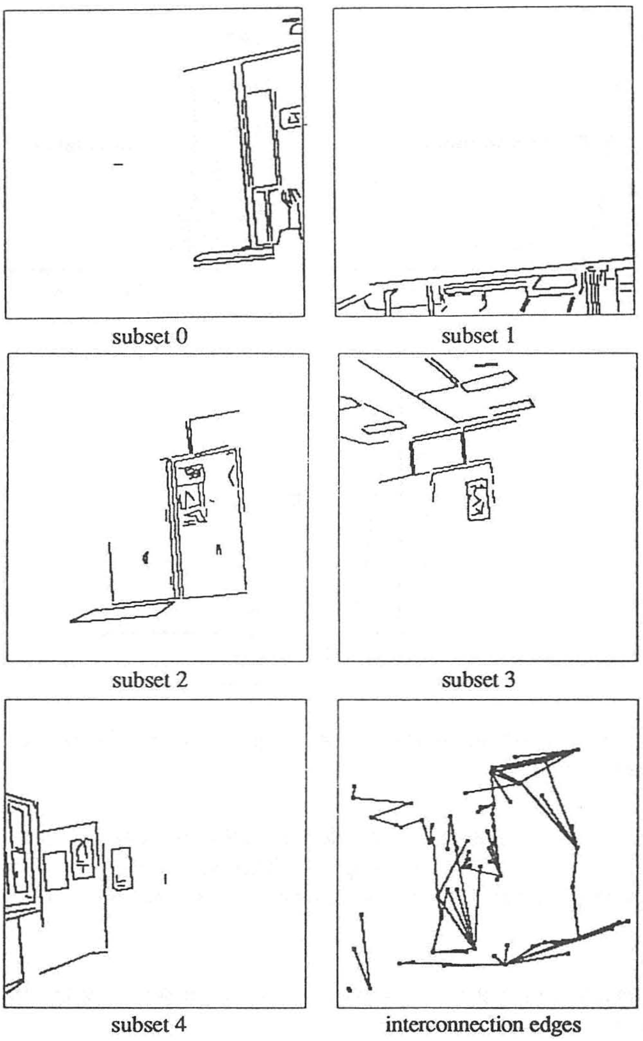

Figure 20: Balanced 5-partitioning with $\alpha = 2$ of the left nonhomogeneous graph of the figure 2 provided by the simulated annealing algorithm. The interconnection cost is 74, the imbalance energy 116,8, and the total energy 74,51. The initial temperature is 40 and the final temperature is 0,172. The number of temperature steps is 127. The distribution of vertices is the following: 63 vertices in subset 0 and 1,68 vertices in subset 2,67 vertices in subset 3 and 62 vertices in subset 4.

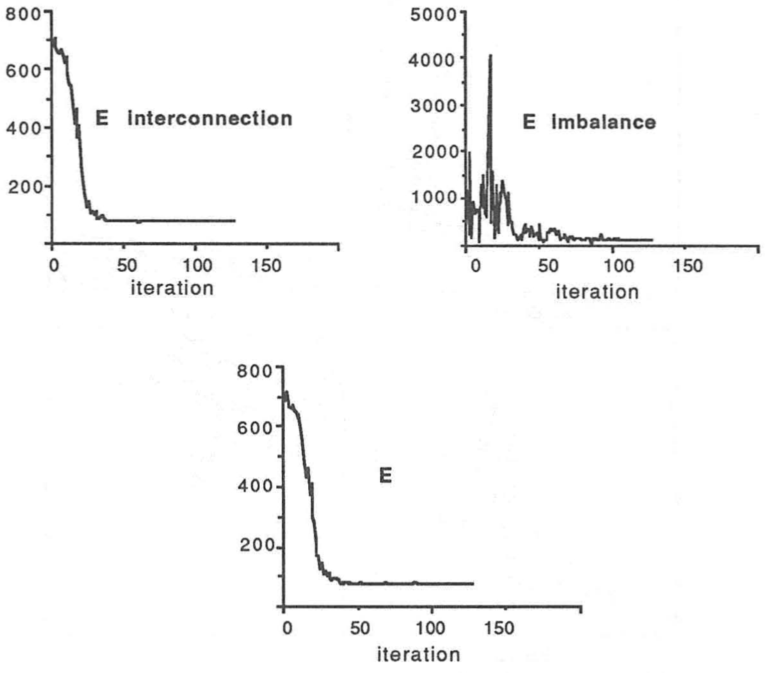

Figure 21: Evolution of the system energies as a function of the temperature step number.

is deterministic and massively parallel. Results are nearly as good as those provided by the simulated annealing algorithm and the CPU time is typically divided by 10 or 20 for simulations on a classical conventional computer (SUN 4-260).

4.3 Partition energy minimization by a network with analog neurons

In this neural network family, the internal state (i.e. the output) of each neuron has an analog internal state.

4.3.1 Hopfield network with analog neurons

One associates a numerical noise to the Boolean transition function previously defined (see section 4.2.1). One would like to mimic the effect of this noise and additionally control the convergence process. So, one considers an Hopfield network with analog neurons as defined in [7]. The new associated

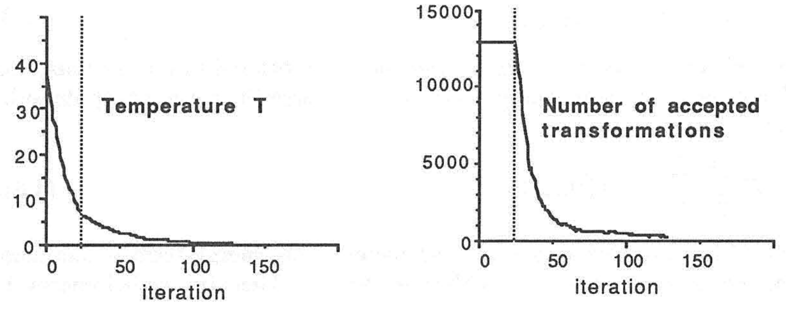

Figure 22: Evolution of the temperature and of the number of accepted transformations as a function of the temperature step number.

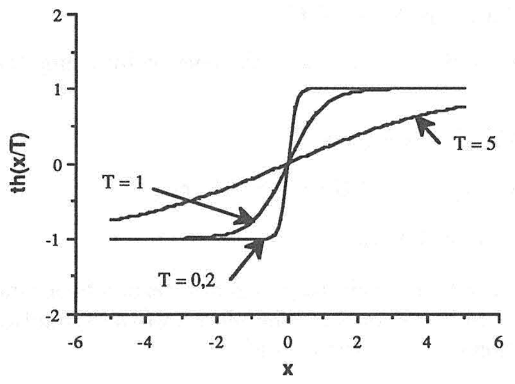

Figure 23: Examples of possible transition functions associated to an analog neuron.

transition function is a sigmoid and depends on a parameter T which mimics the noise. One can take as a transition function (see figure 23):

$$f_T = th(x/T). \tag{4.32}$$

One forces T to tend to 0 during the convergence process. At this limit, we obtain the previous model (see section 4.2.1). As previously, one can show that the Hopfield energy defined in the part 4.2.1 converges to a minimum [7].

A thresholding is made at the end of the process, when one estimates that the network has converged, by using the formula

$$\forall i \in \langle 1, N\rangle, \forall k \in \langle 1, K\rangle,$$
$$V_i^k(t+1) = f_{T\to 0}(V_i^k(t)). \tag{4.33}$$

In this case, one has to add to the energy associated to our optimization problem the additional energy which tends to force the neuron outputs to be +1 or −1:

$$C.\sum_{k=1}^{K}\sum_{i=1}^{N}(1 - V_i^k).(1 + V_i^k) \tag{4.34}$$

where C is a positive adjustable parameter. This energy term is minimum when the neuron outputs are either +1 or −1. Then the global energy to minimize becomes

$$\begin{aligned} E \;=\; & -\frac{1}{2}.\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}[-2.(\lambda - a_{ij}).\delta_{kl} - 2.B.\delta_{ij} + 2.C.\delta_{ij}.\delta_{kl}].V_i^k.V_j^l \\ & - \sum_{k=1}^{K}\sum_{i=1}^{N}[2.B.(2-K) + 4.\lambda.N_k].V_i^k \\ & + B.N.(2-K)^2 + C.N.K. \end{aligned} \tag{4.35}$$

This energy can be written as an Hopfield energy by using the synaptic weights:

$$\forall (i,j) \in \langle 1, N\rangle^2, \forall (k,l) \in \langle 1, K\rangle^2,$$
$$T_{ik,jl} = -2.(\lambda - a_{ij}).\delta_{kl} - 2.B.\delta_{ij} + 2.C.\delta_{ij}.\delta_{kl}, \tag{4.36}$$
$$I_{ik} = 2.B.(2-K) + 4.\lambda.N_k. \tag{4.37}$$

It is equivalent to add an excitatory synaptic term which tends to force every neuron output to be +1 or −1. The matrix of the synaptic weights is symmetrical. Additionally, with the condition

$$C = B + \lambda, \tag{4.38}$$

all the diagonal coefficients of the matrix are 0. Therefore, the system converges to a minimum [7]. With this condition, two parameters have to be fixed: B and λ.

As in the case of an Hopfield network with binary neurons, we have not validated this method on a conventional computer because of the large CPU times which are expected.

In the two types of Hopfield networks (with analog or binary neurons), every neuron is connected to $N-1+K$ other neurons. Therefore the total number of synaptic connections in the network is $N.(N+K-1)$ and thus is proportional to N^2. Consequently a subset number which is small compared to the vertex number does not noticeably influence the running time.

4.3.2 Mean field theory

The main drawback of the simulated annealing is its large running time on a conventional sequential computer (see section 4.2.2). A neural approach coming from statistical mechanics and named mean field theory (MFT) has been developed [17] to solve much more quickly some optimization problems. Here, the data used are scalar entities. We extend this approach to solve optimization problems having a lot of degrees of freedom such as the graph K-partitioning by using the previously defined vectorial entities (equation 4.1). Contrary to the simulated annealing, the convergence process is perfectly deterministic and is controlled by a dynamic system. At every temperature, a solution of this system is directly related to the vertex membership probabilities (between -1 and $+1$) of a subset in the K-partition. We show that this method gives very good results in a smaller CPU time than the one which is necessary in the simulated annealing. Additionally, it is intrinsically massively parallel by nature.

By analogy to physics [19], let us define, for all vertex i and for all subset k,

$$\langle h_i^k \rangle = -2 \sum_{j=1}^{N} (\lambda - a_{ij}).\langle V_j^k \rangle, \tag{4.39}$$

$$h_{\text{ext}}^k = 4.\lambda.N_k \tag{4.40}$$

and

$$\langle H_i^k \rangle = \langle h_i^k \rangle + h_{\text{ext}}^k. \tag{4.41}$$

h_i^k may be considered as the kth component of the field vector created on the vertex i by the other kth spins associated to the graph vertices. h_{ext}^k may be considered as the kth component of the external field in which the system is plunged. Thus H_i^k is the kth component of the total field existing on the vertex i. Then the mean energy of the system (equation 4.14) can be written

$$\langle E \rangle = -\sum_{i=1}^{N} \langle \vec{H}_i \rangle.\langle \vec{V}_i \rangle. \tag{4.42}$$

Let us consider a vertex which is isolated from the others which are supposed to be fixed. Then the mean energy associated to the vertex i is

$$\langle E(\vec{V}_i) \rangle = -\langle \vec{H}_i \rangle.\langle \vec{V}_i \rangle. \tag{4.43}$$

In the spin vector of the vertex i, only one component is $+1$ and the others are -1. We can write the partition function associated to the mean behavior of a vertex as

$$Z = \sum_{\vec{V}} \exp\left\{ -\frac{1}{T}.\langle E(\vec{V}) \rangle \right\} = \sum_{\vec{V}} \exp\left\{ \frac{1}{T}.\langle \vec{H} \rangle.\langle \vec{V} \rangle \right\}. \tag{4.44}$$

In this expression, the configuration of minimum energy are predominant. After some algebra, we obtain

$$Z = \sum_{k=1}^{K} \exp\left\{\frac{1}{T}.\left(\langle H^k\rangle - \sum_{l=1,l\neq k}^{K} \langle H^\ell\rangle\right)\right\}. \tag{4.45}$$

The mean vector of spins associated to a graph vertex has the following kth component:

$$\begin{aligned}\langle V^k\rangle &= \frac{1}{Z}.\left[\exp\left\{\frac{1}{T}.\left(\langle H^k\rangle - \sum_{l=1,l\neq k}^{K}\langle H^l\rangle\right)\right\}\right.\\ &\left.- \sum_{l=1,l\neq k}^{K} \exp\left\{\frac{1}{T}.\left(\langle H^l\rangle - \sum_{m=1,m\neq l}^{K}\langle H^m\rangle\right)\right\}\right].\end{aligned} \tag{4.46}$$

After simplifications, it leads:

$$\langle V^k\rangle = \frac{\exp\left\{\frac{2}{T}.\langle H^k\rangle\right\} - \sum_{l=1,l\neq k}^{K}\exp\left\{\frac{2}{T}.\langle H^l\rangle\right\}}{\sum_{l=1}^{K}\exp\left\{\frac{2}{T}.\langle H^l\rangle\right\}} \tag{4.47}$$

$$= \frac{2}{\sum_{l=1}^{K}\exp\left\{\frac{2}{T}.(\langle H^l\rangle - \langle H^k\rangle)\right\}} - 1 \tag{4.48}$$

The mean field approximation consists in supposing that the field seen by a vertex is the mean field created on this vertex by the other vertices. Then, for all vertex i, we get

$$\langle V_i^k\rangle = \tag{4.49}$$

$$\frac{2}{\sum_{l=1}^{K}\exp\left\{\frac{1}{T}.\left[4.\lambda.(N_\ell - N_k) - 2.\sum_{j=1}^{N}(\lambda - a_{ij}).(\langle V_j^l\rangle - \langle V_j^k\rangle)\right]\right\}} - 1.$$

The solutions of the equation (4.49) can be iteratively obtained thanks to the following equations:

$$\forall i \in \langle 1, N\rangle, \forall k \in \langle 1, K\rangle,$$

$$V_i^{k\ \text{new}} = \tag{4.50}$$

$$\frac{2}{\sum_{l=1}^{K}\exp\left\{\frac{1}{T}.\left[4.\lambda.(N_l - N_k) - 2.\sum_{j=1}^{N}(\lambda - a_{ij}).(V_j^{l\ \text{old}} - V_j^{k\ \text{old}})\right]\right\}} - 1.$$

The desired values are also solutions of the dynamic system:

$$\forall i \in \langle 1, N\rangle, \forall k \in \langle 1, K\rangle, \qquad (4.51)$$

$$\tau.\frac{d\langle V_i^k(t)\rangle}{dt} = -\langle V_i^k(t)\rangle + \frac{2}{\sum_{l=1}^{K} \exp\left\{\frac{1}{T}.\left[4.\lambda.(N_l - N_k) - 2.\sum_{j=1}^{N}(\lambda - a_{ij}).(\langle V_j^l(t)\rangle - \langle V_j^k(t)\rangle)\right]\right\}} - 1.$$

Two running modes are possible. A new step in a synchronous running, every V^k component of each vertex is simultaneously updated by using the other V^k values which have been calculated at the previous step. In the case of an asynchronous running, one calculates the V^k associated only to one node. The V^k values of the other vertices will be calculated in another step. We can logically think that an asynchronous running mode produces best results because the convergence process is less subject to the oscillations which frequently exist in a synchronous running mode. The algorithm is given in appendix C.

Let us make some remarks about this algorithm. One can see that the choice of the final partition is obvious. When one estimates that the system has converged, one chooses for every vertex i the greatest V_i^k among all the positive components V_i^k. The corresponding component has a probability greater than 50% to be $+1$. All the other components are put to -1.

To determine the initial configuration of the system when $N_1 = N_2 = \ldots = N_K$, let us notice that if $V^k(t=0) = 2/K - 1$ for all vertex components, then the components V_i^k are solutions of the dynamic system. Practically, one determines the initial configuration of the system by adding noise on this trivial solution: for instance, the V_i^k are randomly chosen between the two values $(2/K - 1 - 10^{-5}, 2/K - 1 + 10^{-5})$.

We have tested this algorithm in the asynchronous running mode. We notice that the components V_i^k oscillate with a damping during the convergence process. Thus it is our interest to scan the graph vertices a lot of times. The minimum scan number necessary to have a good solution depends on the graph homogeneity: the more nonhomogeneous the graph, the smaller the necessary scan number because the energetic slopes are then more abrupt. Practically, $N/2$ scans are sufficient for nonhomogeneous graphs such as monocular descriptions (see figure 2). As for homogeneous graphs, the system converges in less than N scans. We verify this assertion with the 5-partitioning of the following graphs:

> homogeneous graph and $\alpha = 1$: regular hexagonal network. We show the partition provided provided by the mean field algorithm at two temperatures. In figure 24, the ambient temperature is 2. Figure 25 shows the evolutions in each subset (k fixed) of the components V_i^k as a function of the scan number. The system needs less than N scans to converge. In figure 26 and 27, the ambient temperature is 4. The partition energies of the figure 24 and 26 partitions are comparable but the solutions correspond to different valleys in the energy landscape.

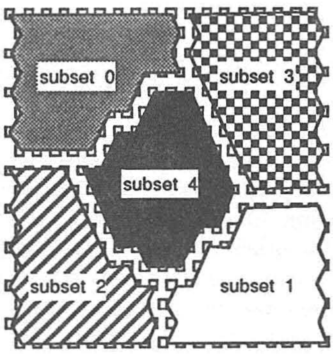

Figure 24: Balanced 5-partitioning with $\alpha = 1$ of a regular hexagonal network of 324 vertices and 901 edges provided by the mean field approximation (edges are not visualized). The ambient temperature is 2. The interconnection cost is 88, the imbalance energy 107, 2 and the total energy 88, 23. The distribution of vertices is the following: 66 vertices in subset 0, 64 vertices in subset 1, 65 vertices in subset 2, 68 vertices in subset 3, 61 vertices in subset 4.

nonhomogeneous graph and $\alpha = 2$: left monocular description of figure 2 (see figures 28 and 29). The system needs less than $N/2$ scans to converge.

Those results are comparable to those given by the simulated annealing but are obtained in a CPU time 10 to 20 times smaller. The interconnection cost given by the generalized Kernighan method are about 20% greater.

The difficulty of this method depends on the choice of the two parameters λ and $T\nabla\lambda$ is chosen without ambiguity (equation 4.15). The choice of the temperature T has not a major influence on the quality of the result when it is chosen in a certain range (between 1 and 4 for graphs having hundreds of vertices). Additionally, one notices that the range of possible temperature increases as the vertex numbers grows.

4.3.3 Mean field annealing

In the mean field approximation algorithm, the temperature is definitively fixed. Another possibility consists in doing an annealing during the convergence process. Consequently the convergence time is reduced: the smaller the temperature, the more rapid the convergence of the previous dynamic system (equation 4.51). Additionally, once the system has converged, the

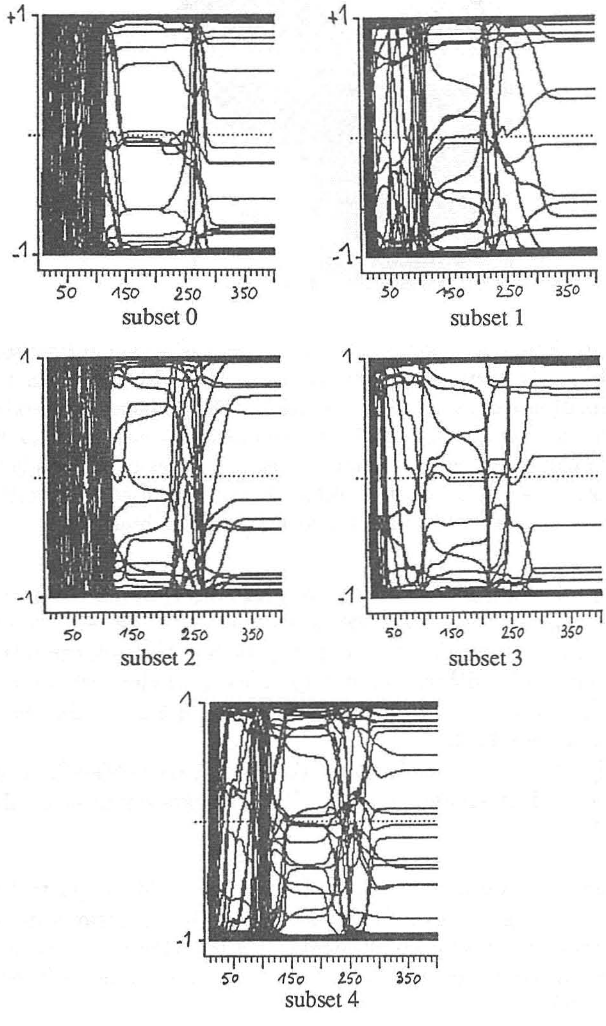

Figure 25: At k fixed, curves giving V_i^k as a function of the scan number.

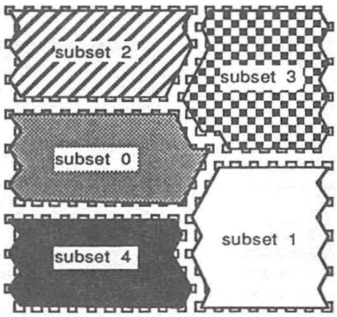

Figure 26: Balanced 5-partitioning with $\alpha = 1$ of a regular hexagonal network of 324 vertices and 901 edges provided by the mean field approximation (edges are not visualized). The ambient temperature is 4. The interconnection cost is 86, the imbalance energy 259, 2 and the total energy 86, 56. The distribution of vertices of vertices is the following: 64 vertices in subset 0, 69 vertices in subset 1, 61 vertices in subset 2, 69 vertices in subset 3, 61 vertices in subset 4.

membership probabilities of a subset are more discriminant than previously (see section 4.3.2): all the V_i^k are forced to tend to $+1$ or -1 when the temperature decreases during the convergence process. The determination of the final partition is made without ambiguity concerning the vertex membership of a subset. The previous algorithm (see section 4.3.2) is slightly modified and is given in appendix D.

Practically, the decreasing factor of the temperature (decT) between two scans must be slightly smaller than 1. We give experimental results in figures 30 and 32:

homogeneous graph, $\alpha = 1$, and decT $= 0,995$: regular hexagonal network (see figure 30). Figure 31 shows the evolutions in each subset (k fixed) of the V_i^k components as a function of the scan number. The system converges much more rapidly than previously (compared to figure 25).

nonhomogeneous graph, $\alpha = 2$, and decT $= 0,995$: left monocular description of figure 2 (see figures 32 and 33).

Those results are comparable to those obtained by using the mean field approximation but the convergence is more rapid and the discretisation which produces the final partition is less ambiguous.

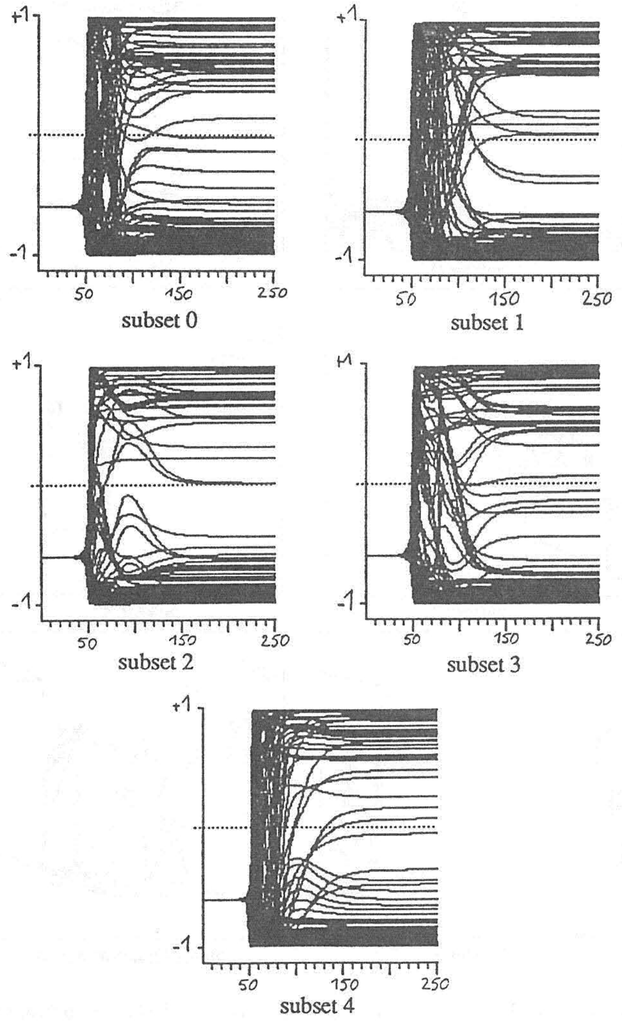

Figure 27: At k fixed, curves giving V_i^k as a function of the scan number.

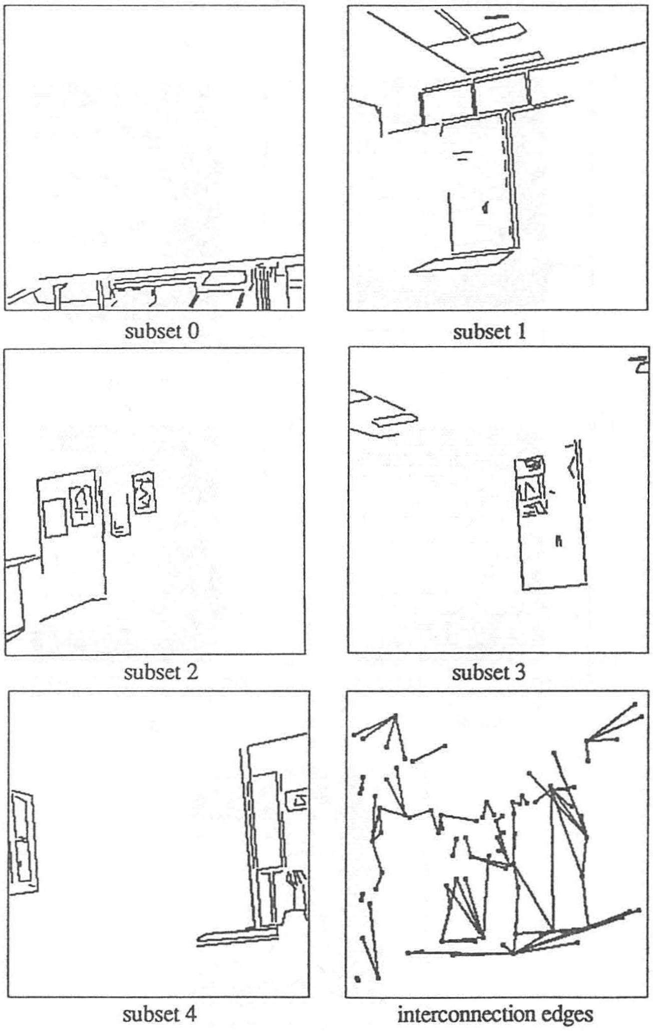

Figure 28: Balanced 5-partitioning with $\alpha = 2$ of the left nonhomogeneous graph of the figure 2 provided by the mean field approximation algorithm. The ambient temperature is 3. The interconnection cost is 85, the imbalance energy 196, 8 and the total energy 85, 86. The distribution of vertices is the following: 63 vertices in subset 0, 66 vertices in subset 1, 63 vertices in subset 2, 61 vertices in subset 3 and 70 vertices in subset 4.

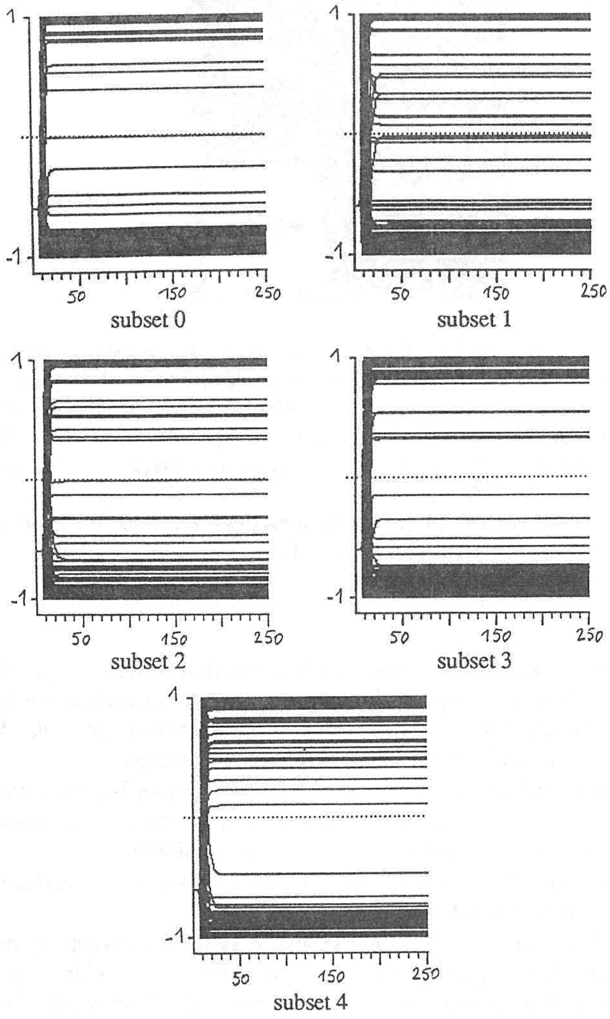

Figure 29: At k fixed, curves giving V_i^k as a function of the scan number.

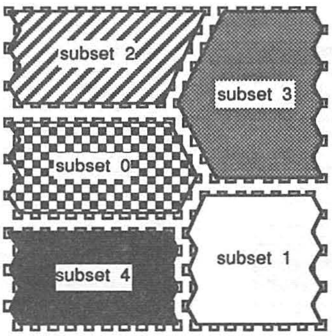

Figure 30: Balanced 5-partitioning with $\alpha = 1$ of a regular hexagonal network of 324 vertices and 901 edges provided by the mean field annealing algorithm (edges are not visualized). The initial temperature is 5 and the decrease coefficient of the temperature is 0, 995. The interconnection cost is 85, the imbalance energy 547, 20 and the total energy 86, 18. The distribution of vertices is the following: 63 vertices in subset 0, 63 vertices in subset 1, 63 vertices in subset 2, 75 vertices in subset 3 and 60 vertices in subset 4.

5. Conclusion

We have shown how a NP-complete combinatorial optimization problem such as the graph K-partitioning can be treated as a minimization problem of a global quadratic energy thanks to the use of vectorial entities. We have proposed several neural methods to minimize this energy.

We have shown how to adapt the synaptic weights between the binary or analog neurons of an Hopfield network so that the system converges to energy minima which are good solutions of our problem.

We have extended the well known simulated annealing procedure (SA) to the use of our vectorial entities.

We have developed a deterministic and massively parallel method using the mean field theory (MFT) to handle our problem. This method, implemented on a conventional computer, gives very good results in a CPU time divided by an order of magnitude 10 to 20 compared to the simulated annealing.

Eventually, in the mean field annealing method (MFA), one makes an annealing during the convergence process of the MFT algorithm. This causes the system to converge more rapidly. Additionally, the final partition is determined with less ambiguity than with the mean field approximation.

Experimental results are given for the SA, MFT and MFA methods.

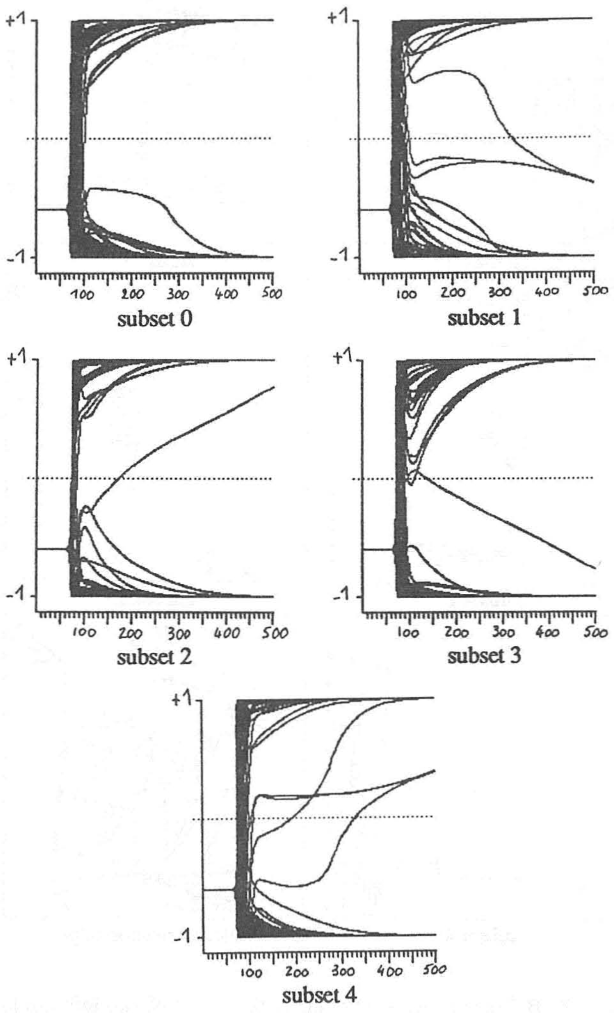

Figure 31: At k fixed, curves giving V_i^k as a function of the scan number.

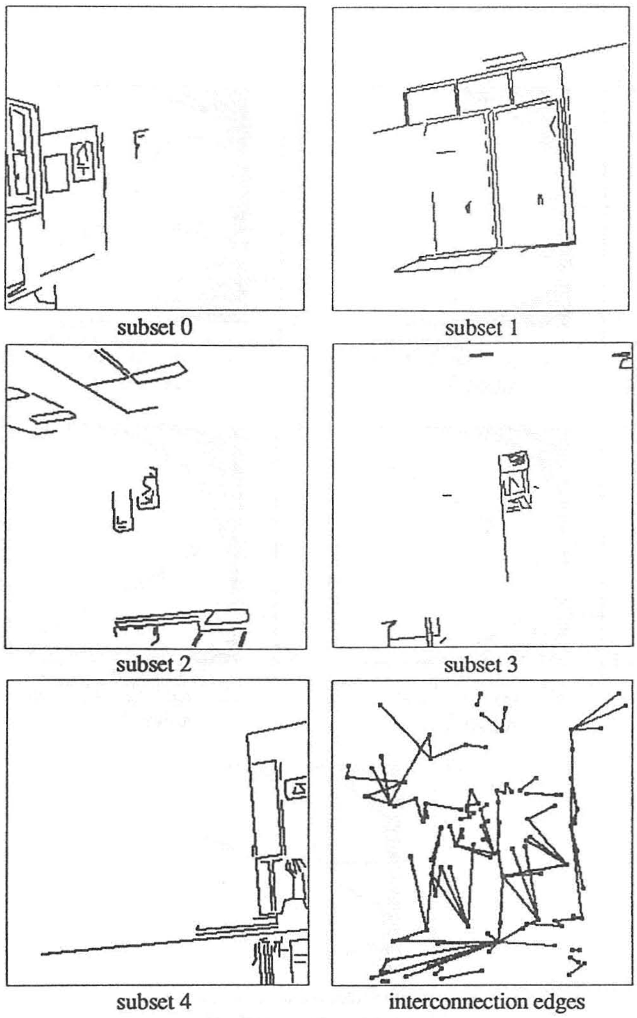

Figure 32: Balanced 5-partitioning with $\alpha = 2$ of the left nonhomogeneous graph of the figure 2 provided by the mean field annealing algorithm. The initial temperature is 5 and the decrease coefficient of the temperature is $0,995$. The interconnection cost is 107, the imbalance energy $532,8$ and the total energy $109,33$. The distribution of vertices is the following: 64 vertices in subset $0,67$ vertices in subset $1,67$ vertices in subset $2,55$ vertices in subset 3 and 70 vertices in subset 4.

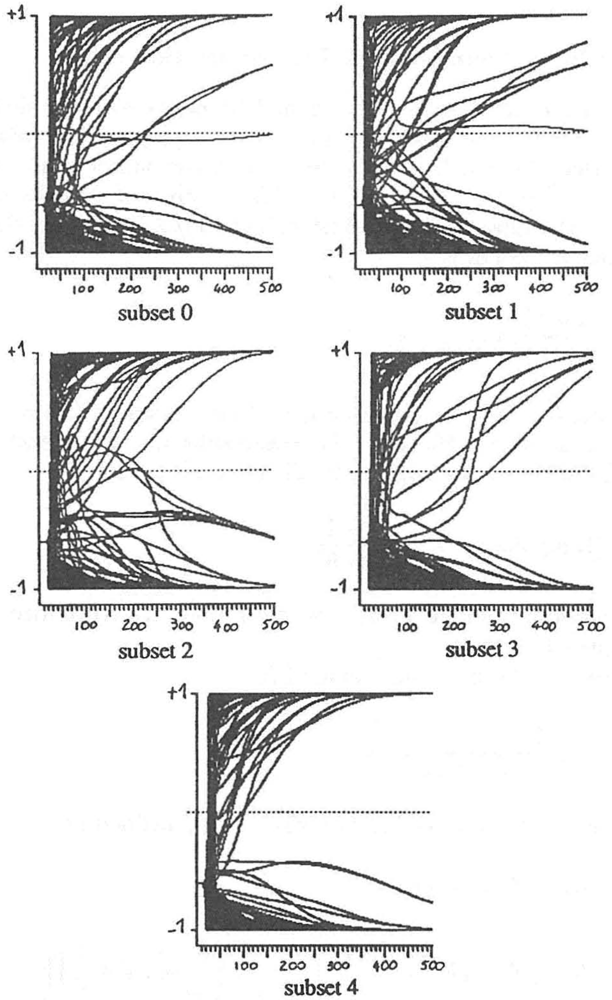

Figure 33: At k fixed, curves giving V_i^k as a function of the scan number.

Appendix A.

In this appendix, upper and lower bounds of the interconnection cost are given as a function of the desired subset number. Additionally, an approximation of the internal density of a subset is developed.

Upper and lower bounds of the interconnection cost

Let us consider a graph of N vertices and M edges with the density $d = 2.M/(N.(N-1))$. One wants to partition this graph in K subsets. Let us suppose that $N(k)$ is the vertex number in the subset k and that $d(k)$ is the internal density in this subset. The number of vertices having its extremities in the subset k is $d(k).N(k).(N(k)-1)/2$. Therefore, the number of interconnection edges is

$$M - \sum_{k=1}^{K} \frac{d(k)}{2}.N(k).[N(k)-1] \tag{A.1}$$

Let us suppose that the partition is perfectly balanced ($N_1 = \ldots = N_K$) and that the density is the same for every subset. In the best case, the number of interconnection edges is 0. Therefore, it leads:

$$\forall k \in \langle 1, K\rangle, d(k) = K.d.\frac{N-1}{N-K} \tag{A.2}$$

with the condition: for all k, $d(k)$ is lower than 1. We notice that $d(k)$ increases when K increases.

Therefore, we obtain a limit value of K:

$$K_{\text{limit}} = E\left[\frac{N}{(N-1).d+1}\right]. \tag{A.3}$$

A lower bound of the interconnection cost is $C_{\min}$ defined by

$$\forall K \leq K_{\text{limit}}, C_{\min}(K) = 0, \tag{A.4}$$

$$\forall K > K_{\text{limit}}, C_{\min}(K) = \frac{N^2.d}{2}.\left[1 - \frac{1}{K}.\left(\frac{d-1}{N.d}.K + \frac{1}{d}\right)\right]. \tag{A.5}$$

In the worst case, for all k, $d(k)$ equals to the graph density. Then an upper bound C_{sup} of the interconnection cost is obtained by replacing $d(k)$ by d in the previous formula. It leads:

$$\forall K \leq N, C_{\text{sup}}(K) = \frac{N^2.d}{2}.\left(1 - \frac{1}{K}\right). \tag{A.6}$$

Approximation of the density in a subset

We suppose that N is much greater than K. Let $M(k)$ be the edge number in the subset k. If the partition is perfectly balanced, we have

$$\forall k \in \langle 1, K\rangle, d(k) = \frac{2.M(k)}{\frac{N}{K}.\left(\frac{N}{K}-1\right)} \approx \frac{2.K^2.M(k)}{N^2}. \tag{A.1}$$

In first approximation, one can take, for all k, $M(k) = M/K$. Therefore:

$$\forall k \in \langle 1, K\rangle, d(k) \approx \frac{2.M}{N^2}.K \approx K.d. \tag{A.2}$$

Appendix B.

In this appendix, we describe the simulated annealing algorithm. We use vectorial entities defined in section 4.1. The algorithm is the following:

1. Get an initial system configuration.
   Construct the associated $(\vec{V_i})_{i\in\langle 1,N\rangle}$.
2. Fix the initial ambient temperature T by using equation 4.31.
   Fix the length of the elementary transformation sequences so as to reach the equilibrium at any temperature $T : L = 100.N.(K-1)$.
3. Get initial number of accepted transformations at this temperature: NT accept $= 0$.
   Repeat L times:
   (a) Pick at random a vertex i of the graph (this vertex is in the subset $k : V_i^k = 1$).
   (b) Pick at random a subset l which is different from k.
   (c) Calculate the energy variation associated to the move of the vertex i from the subset k to the subset l by using equation 4.29.
   (d) If the energy decreases:
      i. The elementary transformation is accepted: NTaccept $\rightarrow$ NTaccept $+ 1$.
      ii. Operate the transformation: $V_i^k = -1$ et $V_i^l = 1$.
   (e) If the energy increases, then the elementary transformation is accepted with a probability given by equation 4.30.
   (f) If NTaccept $= L/10$, then consider that the equilibrium is reached at T: stop (go to step 4.).
4. If NTaccept $= L/10$, update the ambient temperature ($T_{\text{new}} = 0,93T_{\text{old}}$) and go to step 3.
   If NTaccept is between N and $L/10$, the system is freezing, update the ambient temperature ($T_{\text{new}} = 0,965T_{\text{old}}$) and go to step 3.
   If NTaccept $< N$ (the system is frozen), stop: the solution (final K-partition) is obtained.

Appendix C.

In this appendix, we describe the mean field approximation algorithm. We use vectorial entities defined in section 4.1. The algorithm is the following:

1. Fix the running mode:

   synchronous $\rightarrow$ fct $= 0$,

   asynchronous $\rightarrow$ fct $= 1$.

   Fix the temperature T.

   Fix the scan number of the graph vertices: Nbscan.

   Get, for all i and k, an initial value V_i^k randomly chosen between the values $(2/K - 1 - 10^{-5}, 2/K - 1 + 10^{-5})$.

2. Repeat Nbscan times:
   (a) Randomly scan the graph vertices in such a way that every vertex is updated once.
      i. Update every vertex seen in the scan:
         A. Calculate, for all k, $V_i^{k\ \text{new}}$ (equation 4.50).
         B. If fct $= 1$ (asynchronous running mode), update for all k: $V_i^{k\ \text{old}} = V_i^{k\ \text{new}}$.

   (b) If fct $= 0$ (synchronous running mode), update for all vertex i and for all subset k: $V_i^{k\ \text{old}} = V_i^{k\ \text{new}}$.

3. (a) Test if the system has converged into a configuration different from the initial one.

   (b) If the system has not converged, either the temperature T is too high or Nbscan is too small. Go to step 1.

   (c) If the system has converged, for all graph vertex i:
      i. Determine k such that V_i^k is the greater.
      ii. Do $V_i^k = 1$ and, for all $\ell \neq k$, $V_i^l = -1$ $\rightarrow$ the vertex i is in the subset k.

Appendix D.

In this appendix, we describe the mean field annealing algorithm. We use vectorial entities defined in section 4.1. The algorithm is the following:

1. Fix the running mode:

   synchronous $\rightarrow fct = 0$

   asynchronous $\rightarrow fct = 1$

   Fix the initial temperature $T = T_0$.

   Fix the scan number of the graph vertices: Nbscan.

   Fix the decreasing coefficient of the temperature between two consecutive scans: decT. Get, for all i and k, an initial value V_i^k randomly chosen between the values $(2/K - 1 - 10^{-5}, K - 1 + 10^{-5})$.

2. Repeat Nbscan times:

   (a) Randomly scan the graph vertices in such a way that every vertex is updated once.

      i. Update every vertex seen in the scan: (1) Calculate, for all k, $V^{k \text{ new}}$ (equation 4.50). (2) If fct $= 1$ (asynchronous running mode), update for all k: $V_i^{k \text{ old}} = V_i^{k \text{ new}}$.

   (b) If fct $= 0$ (synchronous running mode), update for all vertex i and for all subset k: $V_i^{k \text{ old}} = V_i^{k \text{ new}}$.

   (c) $T \rightarrow \text{decT} * T$.

3. (a) Test if the system has converged into a configuration different from the initial one.

   (b) If the system has not converged, Nbscan is too small. Go to step 1.

   (c) If the system has not converged, for all graph vertex i:

      i. Determine k such that V_i^k is the greater.

      ii. Do $V_i^k = 1$ and, for all $\ell \neq k$, $V_i^1 = -1 \rightarrow$ the vertex i is in the subset k.

Acknowledgments

The authors would like to thank Dr. R. Horaud and Dr. T. Skordas for enlightening discussions.

References

[1] Ayache and Faverjon, "Efficient registration of stereo images by matching graph descriptions of edge segments," *International Journal of Computer Vision*, **1(2)** (1987).

[2] S. Bokhari, *Assignment Problems in Parallel and Distributed Computing* (Kluwer Academic Publishers, 1987).

[3] M. Burstein, "Algorithms for partitioning of VLSI networks," *IBM Technical Disclosure Bulletin*, **25(11A)** (1983).

[4] W.E. Donath and A.J. Hoffman, "Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices," *IBM Technical Disclosure Bulletin*, **15(3)** (1972).

[5] M.R. Garey and D.S. Johnson, *Computers and Intractability* (W.H. Freeman and Company, New York, 1979).

[6] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6(6)** (1984) 721–741.

[7] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-states neurons," *Proc. Natl. Acad. Sci. USA*, **81** (1984).

[8] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, **79** (1982).

[9] R. Horaud and T. Skordas, "Stereo-correspondence through feature grouping and maximal cliques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11(11)** (1989).

[10] B. Kernighan, *Some Graph Partitioning Problems Related to Program Segmentation* (Princeton University, Ph.D., 1969).

[11] B. Kernighan and S. Lin, "An efficient Heuristic procedure for partitioning graphs," *The Bell System Technical Journal* (February, 1970).

[12] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing" IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1982.

[13] E.L. Lawler, "Electrical assemblies with a minimum number of interconnections," *IEEE Transactions Electronic Computers*, **EC-11** (1962).

[14] J.A. Lukes, *Combinatorial Solution to Partitioning Problems* (Stanford Electronics Laboratories Technical Report No. 32, 1972).

[15] J.A. Lukes, "Combinatorial solution to the partitioning of general graphs," *IBM J. Res. & Dev. (USA)*, **19(2)** (1975) 170–180.

[16] "Procedure for partitioning the nodes of a graph," *IBM Technical Disclosure Bulletin*, **28(9)** (1986) 4030-4034.

[17] C. Peterson and J. Anderson, "Neural networks and NP-complete optimization problems: A performance study on the graph bisection problem," *Complex Systems*, **2** (1988).

[18] G. Rao, H. Stone, and T. Hu, Assignment of tasks in a processor system with limited memory," *IEEE Conf. on "Neural Information Processing Systems - Natural and Synthetic"* (Denver, Colorado, November 8–12, 1987).

[19] H.E. Stanley, "Introduction to phases transitions and critical phenomena," *The International Series on Monographs on Physics* (Oxford University Press, 1971).

[20] H. Stone, "Multiprocessor scheduling with the aid of network flow algorithms," *IEEE Computer*, **16(1)** (1983).

[21] G.A. Tagliarini and E.W. Page, "A neural-network solution to the concentrator assignment problem," *IEEE Conf. on "Neural Information Processing Systems - Natural and Synthetic"* (Denver, Colorado, November 8–12, 1987).

[22] D.W. Tank and J.J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Transactions on Circuits & Systems*, **CAS-33(5)** (1986).