Massively Parallel Image Segmentation on the Connection Machine

Marc Berthod*, Gérard Giraudon*, Jean Paul Stromboni+
INRIA, BP.93, 06902 Sophia Antipolis France
+I3S, URA 1376 CNRS, 41, Boulevard Napoléon III, 06041 Nice France

Abstract

This paper is devoted to a new deterministic and massively parallel algorithm for combinatorial optimization in a Markov Random Field. First, the a posteriori probability of a tentative labeling, defined in terms of a Markov Random Field is generalized to continuous labelings. This merit function of probabilistic vectors is then convexified by changing its domain. Global optimization is performed, and the maximum is tracked down while the original domain is restaured. We analyse in details the parallel implementation of this algorithm on the CM2, in terms of speed efficiency, memory and communication requirements. Comparison with other classical algorithms is made on a contextual pixel classification problem.

1 Introduction

Since the seminal paper of Geman [6], Markov Random Fields (M.R.F.) many heuristic algorithms have been proposed for finding the Maximum a Posteriori Mode (M.A.P.) in low-level vision problems: iterated conditional modes [2], simulated annealing [6], dynamic programming [4] etc ...

Starting from Relaxation Labeling [5], we have proposed in a recent paper [8] a Deterministic Pseudo Annealing (D.P.A.) algorithm, a variation on annealing. The basic idea is to extend the probability of a labeling (a function defined on a discrete set) to a merit function defined on continuous labelings (a subset of $\mathcal{R}^{\mathcal{N}}$): a polynomial with non-negative coefficients. The only extrema of this function, under suitable constraints, occur for discrete labelings. D.P.A. consists of changing the constraints so as to convexify this function, find its unique global maximum, and then track down the solution, by a continuation method, until the original constraints are restored, and a discrete labeling can be obtained.

We describe here the parallelization and an implementation on the Connection Machine, with an application to pixel quantization.

2 Deterministic-Pseudo-Annealing : the Method

In this section, we present briefly DPA algorithm (see [8] for more details). Let $\mathcal{S} = S_i,\ 1 \leq i \leq N$ be a set of sites (pixels in this paper), each of which may take any label from 1 to M. A global discrete labeling L assigns one label L_i to each site S_i.

We assume that the a priori probability of L is modeled by a M.R.F. Let $Y = (y_1 \dots y_N)^t$ be the vector of observations on the sites. Using reasonable independance hypotheses on the noise, the a posteriori probability $P(L/Y)$ (using Bayes theorem) follows a Gibbs distribution, with the same associated graph G.

Let V_i be the set of sites connected to S_i, in G. $\mathcal{C}$ is the set of all the cliques c of G. Also $C_i = \{c : S_i \in c\}$. The number of sites in the clique is its degree : $\deg(c)$, and $\deg(G) = \max_{c \in \mathcal{C}} \deg(c)$.

Let L_c be the restriction of L to the sites of c,and W_{cL} the clique potential for $c \in \mathcal{C}$ and $L \in \mathcal{L}$ ($\mathcal{L}$ is the set of the M^N discrete labelings). Following Hammersley-Clifford, W_{cL} is proportional to a joint probability on c. The M.A.P. can be cast in the following way:

$$L^* = \max_{L} \sum_{c \in \mathcal{C}} W_{cL} \tag{1}$$

Now, let $\mathcal{R}^{\mathcal{NM}} \xrightarrow{f} \mathcal{R}$ be defined by:

$$f(X) = \sum_{c \in \mathcal{C}} \sum_{l_c \in L_c} W_{cl} \prod_{j=1}^{\deg(c)} x_{c_j, l_{c_j}} \tag{2}$$

where c_j denotes the j^{th} site of clique c, and l_{c_j} the label assigned to this site by l_c. f is a polynomial in the $x_{i,k}$'s, linear with any $x_{i,k}$; its degree is $\deg(G)$.

Let us now restrict X to $\mathcal{P}^{NM}$, defined by $\forall i,k \ : \ x_{i,k} \geq 0 \ \ \& \ \ \forall i \ : \sum_{k=1}^{M} x_{i,k} = 1$. It turns out that, generically, if X^* is a critical point of f on $\mathcal{P}^{NM}$ then it is on the border, i.e.: $\forall i, \exists k \ : \ x^*_{ik} = 1$, and thus it directly yields a discrete labeling. Thus, the absolute maximum of f (which has many local maxima) yields the solution to our problem.

Now, let $Q^{NM,d}$ be the compact subset of $\mathcal{R}^{NM}$ defined by:

$$\forall i,k \ : x_{i,k} \geq 0 \quad \& \quad \forall i \ : \sum_{k=1}^{M} x_{i,k}^d = 1$$

It can be proven that f admits a unique maximum on $Q^{NM,d}$. When $d = 2$ and $N = 1$, this reduces to Perron-Frobenius theorem on non-negative matrices.

Maximization is performed, starting from some X^0, by applying:

$$X^{n+1} \simeq (\bigtriangledown f(X^n))^{\frac{1}{d-1}} \tag{3}$$

This simply means that, at each iteration, we select on the pseudo-sphere of degree d the point where the normal is parallel to the gradient of f. When $d = 2$, this is the well known iterative power method for finding eigenvectors. Obviously, the only stable point is singular, and thus is the maximum we are looking for. We have only proved experimentally that the algorithm does converge very fast to this maximum.

This procedure, already suggested in [1] yields a maximum which, as in the case $d = 2$, is inside $Q^{NM,d}$ (degeneracies apart), and thus does *not* yield a discrete labeling. So we actually track down the solution, maximizing f on successive $Q^{NM,\beta}$'s, with β decreasing from d to 1, starting from the last maximum.

This iterative decrease of β can be compared, up to a point to a cooling schedule, or better to a Graduated Non-Convexity strategy [3]. It also shares some common flavor with mean-field approximation [9].

Obviously, this scheme is not necessarily optimal (simple counterexamples can be designed), but a very good solution is usually reached on real problems.

Besides, let us notice that, though shifting the coefficients does not change the discrete problem nor the maximization problem on $\mathcal{P}^{NM}$, it changes it on $Q^{NM,d}$, and thus there is no guarantee that the same solution is reached.

Finally, experiments have shown that the speed with which β is decreased is not crucial : typically, 5 to 10 steps are enough to go from 2 to 1.

3 An Application to Image Segmentation

The problem at hand here is to quantize an image in such a way that areas with a given gray level are nicely connected. So we expect isolated pixels, or small isolated areas with a grey level different from their context to be eliminated. Here, the units (or sites) are the pixels, and the classes (or labels) are the quantized grey-levels (typically 2 to 5). The result will be a segmented image into 2 to 5 classes.

The World Model (cliques of order 2) favours similar classes for neighbouring pixels, and penalizes different labels. For example, if two neighbouring sites have the same label, the energy is 0, else it is -1. This actually means that the ratio between the a priori probability of having the same labels, to the a priori probability of having different labels is $exp(1)$ (i.e. approx. 2.7).

We have used test images of miscellaneous types. In this paper just two are presented. The first one is a mineral image (512*512*8). A result is shown in Figure 2 for 5 classes segmentation. The second one is an indoor scene (512*512*8). A result is shown in Figure 3 for 5 classes. In both cases, the values of center classes are $0, a, 2a, 3a, 4a$ where $a = 63.75$ and with $\sigma = a/2$.

The other example is a synthesized (128*128*8) noisy chessboard, obtained by corrupting a binary picture by noise with a $-5dB$ noise (Figure 4). It is used to illustrate comparison with other methods to segment image in two classes. The challenger methods are : mean and median filtering, anisotropic diffusion filtering [7] and Graduated Non Convexity (or GNC) [3]. The range of the computational complexity of these methods is already quite large, although we have not compared our method with the heaviest method, simulated annealing, on this application. For every method, parameters are optimized to obtain the best result (window range, iteration, thresholding, etc...). In our method, the class mean values are 0 and 255 with $\sigma = 255$.

The first three methods are implemented in C on a-Sun 4 workstation and the fourth in *Lisp on the CM.

The performances are established in two ways :

- a) objective performances essentially based on the correct classification (Figure 1).
- b) subjective performances based on degradation of visual aspect, as shown in Image 3 where just best relaxation and GNC results are presented.

DPA algorithm has the best of objective and subjective performance. It took 69 iterations and 0.8 seconds on the Connection Machine. For mean filtering, the best objective performance is obtained with window size 3x3 but subjectively, a 9x9 window looks better. The discussion is similar for median filtering. In both cases, the result is obtained in a few seconds with a Sun 4 station. Anisotropic diffusion results are obtained in a few minutes with a Sun 4. To wind up, GNC best result also on Image 3 asked 250 iterations and 10 seconds on the Connection Machine.

For objective performance in Figure 1, three indicators were defined for two classes, the *win*, *alert* and *error* ratios, as follows :

$$win = \frac{Number\ of\ white\ pixels\ classed\ white}{Number\ of\ input\ white\ pixels}$$

$$alert = \frac{Number\ of\ white\ pixels\ classed\ black}{Number\ of\ input\ white\ pixels}$$

$$error = \frac{Number\ of\ initially\ black\ pixels\ classed\ white}{Number\ of\ initially\ black\ pixels}$$

The results may then be summarized as follows :

- average and median filtering lead to poorer results, but they are very simple algorithms,
- anisotropic filtering gives intermediate results for intermediate complexity
- GNC, which is the most complex method, and the present relaxation algorithm, win the challenge and obtain similar results. The relaxation turns out to be faster and slightly more performant on the present example. It offers the best trade-off between performance and complexity.

Considering subjective performance analysis, choice is much more difficult. Anyway, it appears that even with a very simple method like smoothing with average, subjective result remains satisfactory. Choice will then depend on the application.

4 Algorithm implementation

4.1 Parameters for parallelization

Many parameters may be tuned in the algorithm. We have tested the following :

- picture dimensions and number C of classes
- pixel neighbourhood (4 or 8)
- stabilization threshold *t1* and stop test threshold *t2* on the number of non stabilized hypotheses (*t1*= 0.1 or 0.001, and *t2*= 0.1 percent of all the hypotheses or 0 percent)
- initial labeling from pixel intensity values (discontinuous initialization law using thresholds, or continuous linear or gaussian laws)
- local rule for neigbours interactions, defined in a local interaction matrix identical for all the pixels in the geometry (identity matrix, with or without coupling terms, interclass reinforcement, or inhibition)
- extraction of final result from equilibrium coefficients values may be continuous or threshold based. From theory, and confirmed by experiment, equilibrium values for hypotheses are close to 1 or 0 and pixel class is chosen without ambiguity.

Then, the whole process may be continuous, from input picture to classified pixels, with gaussian initial labeling law and continuous class extraction (particularly with a final β of one).

We have analysed different aspects of our implementation : noise sensibility and comparison with some other classical methods of segmentation, convergency behavior, influence of initial labelling law and of local interaction rules, iteration speed.

4.2 Parallel implementation and Parallelization Performance

The Connection Machine CM2 is used for the execution of D.P.A. algorithm, with possibly one or two sequencers on site (8192 or 16384 one-bit processors connected on a hypercube network). Programming level is C/Paris, akin to assembler. The plain parallel algorithm formulation suggests a parallelization *by inspection*. Allocating one machine site to every pixel leads to regular and local data exchanges between sites and identical site computation, at once the same load and operations. D.P.A. algorithm actually well fits the machine in SIMD processing mode and NEWS communication mode for transfer. Load balance can be predicted, with speedup only bounded by the communication delays and Communication Computation Ratio.

In the general case, C labels ought to be associated to every pixel (floating numbers comprised between 0 and 1). The next label value in pixel i is computed from K connected pixels or neighbours labels (here north, south, east and west of pixel i in the image). All data concerning pixel i is mapped in one processor, as shown in Figure 5, and the resulting amount of memory needed for every pixel in this implementation is: $M =$

$B + 3CF + KCF + KC^2F = 1 + 12C + 4KC + 4KC^2$ bytes. If the number $D \times D$ of pixels is greater than the number P of CM2 processors, each processor will successively manage *vpr* pixels:

$vpr = D \times D/P$ if integer, or int$((D \times D/P) + 1)$ if not.

For *Chessboard* shown in Figure 4, $D \times D = 128 \times 128$ pixels, with 2 classes, the 4 NEWS neighbours, and $P = 8192$ processors, *vpr* is 2, the memory required is 242 bytes per processor, and the total CM2 memory used approaches 2 Megabytes. It increases as K, D^2 and C^2.

The D.P.A. includes two main parts. The first one initializes labels from the image loaded. The second one, iterative relaxation process, has 4 phases: label diffusion, label accumulation, vector label normalization and a check stop condition. After these two parts, thresholding can be applied for classification improvement (decision part).

From the execution times recorded for different images, initialization roughly consumes one iteration duration. If used, the decision part is faster because requiring no transfer. Thus the approximate formula $T = N \times t_{iterate} + t_{initialize}$ stands for the duration of N iterations. Measured with one sequencer, the mean iteration time is approximately linear versus the number of classes and the image size:

$$t_{iterate} \simeq 0.01 \times \frac{C}{2} \times (\frac{D}{128})^2 \ (in\ seconds)$$

In order to evaluate the actual parallelization performance, efficiency and speed-up gained, the particular analysis of the list of the transfer and processing operations involved in one iteration can be conducted. Associated with CM2 Instruction Times, as obtained from the Technical Manual for *vpr* 1 and *vpr* 16, the list leads to an evaluation of the D.P.A. execution time, parallel efficiency, and speed-up.

For one iteration (four procedure calls), the following amount of operations is found ($D \times D$ image, P processors and $K \times P$ communication links, the "$t_{operation}$"s are some parallel operations drawn from the PARIS Parallel Instruction Set of the CM2) :

1. $P \times t_{communicate} = D(D-1)CK \times t_{transfer}$, communication of neighbour labels,
2. $P \times t_{accumulate} = D^2C(2t_{multiply} + (KC-1)t_{multadd})$, accumulation of influences,
3. $P \times t_{normalize} = D^2((C+1)t_{power} + Ct_{divide} + (C-1)t_{add})$, normalization of label vector
4. $P \times t_{compare} = D^2C(t_{substract} + t_{negate} + t_{greaterthan}) + t_{greaterthan}$, checking the stop condition,

and $t^P_{iterate} = t_{communicate} + t_{accumulate} + t_{normalize} + t_{compare} = t_P$.

Then, $t_{accumulate}$ grows as C^2 and the other durations as C. With only one processor, and no need for transfers on the hypercube, this would have become:

$t^1_{iterate} = P \times t_{accumulate} + P \times t_{normalize} + P \times t_{compare} = P \times t_1$

Parallelization efficiency is then defined as:

$$E = \frac{SpeedUp}{P} = \frac{P \times t_1}{t_P} \times \frac{1}{P} = \frac{t_1}{t_{communicate} + t_1} = \frac{1}{1 + \frac{t_{communicate}}{t_1}} = \frac{1}{1+e}$$

Coefficient e is a function of K, C, and D, and some CM2 execution times extracted from PARIS Manual. Using *vpr* 1 Time Table in the Manual, which means $D^2 \leq P$, with $K = 4$ and $C = 2$, leads to the following issues (times in μseconds):

$t_{communicate}$	$t_{accumulate}$	$t_{normalize}$	$t_{compare}$	$t_{iterate}$	E
1270	1188	3128	194	5780	0.78

Using then *vpr* 16 data in the Manual gives a different $t_{iterate}$ = 60822 and a better efficiency of 0.85. From these two results, and linear variation hypothesis of execution times with *vpr* (CM2 Manual), it comes :

$t_{iterate}(vpr) = 2111 + vpr \times 3669$ μseconds (± 10 %).

Prediction for *vpr* 2 is then 9.5 milliseconds ± 10 %, 31.7 milliseconds for *vpr* 8 and 0.12 seconds for *vpr* 32, according to the experimental issues obtained from various images. Considering also the model prediction of the efficiency, related to processing speed-up gained from parallelization, it only depends for a given image size on the number of classes specified. For *vpr* 1:

C classes	2	3	4	5
Efficiency E	0.78	0.79	0.80	0.82

5 Conclusion

In this paper, a new deterministic algorithm for combinatorial optimization in MRF is presented. We insist on the implementation point of view on a massively parallel architecture. The parallelization on the CM of the regular scheme for data transfer and accumulation drawn from the algorithm formulation is efficient, as could be a priori supposed. Efficiency, as determined from the evaluation of the amount of necessary communications, lies from 75 to 85 percent. It should be emphasized that a 80 percent efficiency ratio means a 6554 processors equivalent machine (1638 CM processors 'wasted' among the 8192 processors used). Then, the large amount of processing units in the CM leads to an important loss of equivalent processors for even small efficiency decrements. Any improvement in communication speed results in better use of actual processing resources.

The execution time of the algorithm is relatively small. For picture *Desk* (512×512), using 8192 CM processors, 5 classes relaxation, and 0.1 percent stop threshold, one initialization step and 22 iterations are needed, i.e. almost $0.37 + 0.313 * 22 = 7.26$ seconds; this time becomes 2.2 seconds for 2 classes. For synthetic picture (128x128) *Chessboard*, 10 iterations i.e. .1 seconds are needed The execution time could be lowered using the fact that diagonal local matrix proved usually efficient which allows to reduce the number of floating operations in the program.

Concerning the basic algorithm performance for picture segmentation, it has been pointed out that the initial labelling should be determined from a preliminary picture analysis, from pixel intensity repartition analysis for instance. Also, n-class relaxation using moving centers for automatic scan of the whole pixel intensity interval should be considered.

These and many other algorithm adaptations are left to further study.

Acknowledgments: The authors would like to thank J.Zerubia and F.Ployette for their contribution in the GNC result.

References

[1] M. Berthod. Definition of a consistent labeling as a global extremum. In *Proc. ICPR6*, pages 339–341, Munich, 1982.

[2] J. Besag. On the statistical analysis of dirty pictures. *Jl. Roy. Statis. Soc. B.*, 1986.

[3] A. Blake. Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. *IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol 11, pp 2-12*, 1989.

[4] H. Derin, H. Elliott, R. Cristi, and D. Geman. Bayes smoothing algorithms for segmentation of binary images modeled by markov random fields. *IEEE trans. on Pattern analysis and mach. intel., Vol 6*, 1984.

[5] O. Faugeras and M. Berthod. Improving Consistency and Reducing Ambiguity in Stochastic Labeling: an Optimization Approach. *IEEE Transactions on PAMI*, 4:412–423, 1981.

[6] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

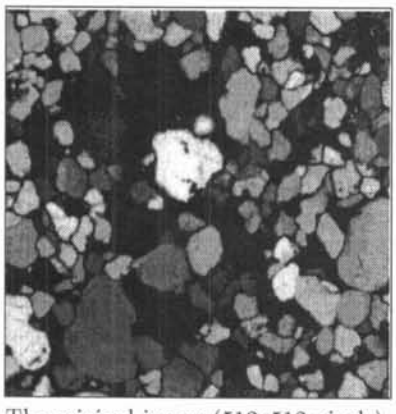
[7] P. S. Marc, J. Chen, and G. Medioni. Adaptative Smoothing : A general tool for early vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(6):514–529, 1991.

[8] M.Berthod, G.Giraudon, and J.P.Stromboni. Determistic Pseudo-Annealing : a Suboptimal Scheme for optimization in Markov Random Field. An application to pixel classification. In *Proc. ECCV*, Santa Margherita Ligure Italy, May 1992.

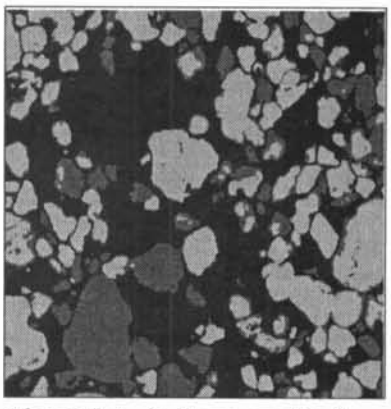
[9] J. Zerubia and R. Chellappa. Mean field approximation using compound Gauss-Markov Random field for edge detection and image restoration. *Proc. ICASSP, Albuquerque, USA*, 1990.

Method	win ratio	alert ratio	error ratio
best relaxation	0.99	0.01	0.01
Averaging 3x3	0.89	0.11	0.06
Averaging 9x9	0.83	0.17	0.02
Median 3x3	0.91	0.09	0.37
Median 9x9	0.84	0.16	0.07
Medioni (1 iter.)	0.92	0.08	0.06
Medioni (50 iter.)	0.90	0.10	0.04
best GNC	0.98	0.02	0.02

Figure 1: Objective performance comparison for some other methods

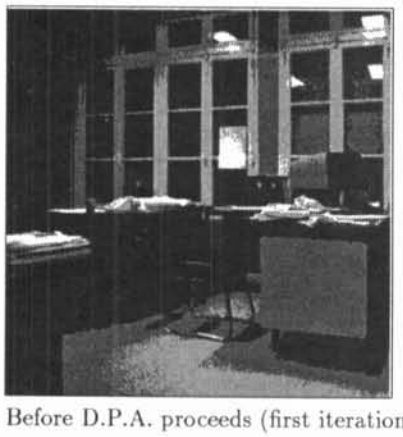


The original image (512x512 pixels)

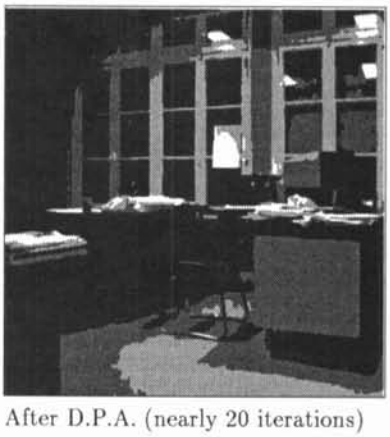


After D.P.A. (5 classes specified)

Figure 2: Example of D.P.A. application to a mineral image

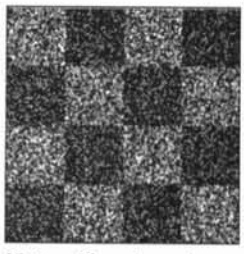


Before D.P.A. proceeds (first iteration)

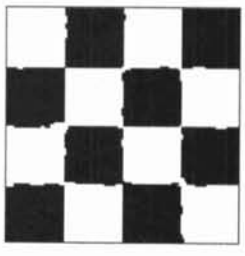


After D.P.A. (nearly 20 iterations)

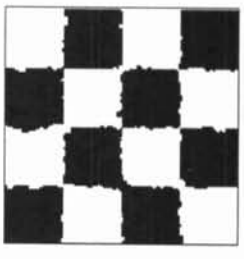
Figure 3: The effect of D.P.A. application to an indoor scene (512x512 pixels, 256 grey levels)



Noisy Chessboard
(-5dB S/N ratio)



Best D.P.A. result



Best G.N.C. result

Figure 4: Visual comparison between G.N.C and D.P.A.

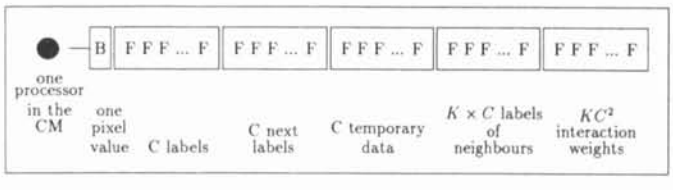


Figure 5: Data area for one pixel (Field sizes: B= 8 bits and F= 32 bits)