Decision Trees for Fast Thinning Algorithms

Costantino Grana, Daniele Borghesani, Rita Cucchiara
Dipartimento di Ingegneria dell'Informazione
Universit degli Studi di Modena e Reggio Emilia, Italy
{*name.surname*}*@unimore.it*

Abstract

We propose a new efficient approach for neighborhood exploration, optimized with decision tables and decision trees, suitable for local algorithms in image processing. In this work, it is employed to speed up two widely used thinning techniques. The performance gain is shown over a large freely available dataset of scanned document images.

1. Introduction

Thinning is a fundamental algorithm, often used in many computer vision tasks, such as document images understanding and OCR.

A lot of algorithms have been detailed in literature to solve the problem, typically in a sequential or parallel fashion (according to the classification proposed by [6]). Parallel techniques in particular gather most attention in literature. One the most famous used algorithm has been proposed by Zhang and Suen [11]. This iteratively runs two subiterations to remove pixels. Holt *et al.* [5] proposed an improvement on this technique which only requires a single iteration, but which requires examining a larger neighborhood, from 3×3 to 4×4. This algorithm requires less iterations, but the need to access more pixels makes it slower when implemented on sequential machines [4]. Moreover Chen and Hsu [1] improved the original Zhang and Suen algorithm and proposed a look up table solution to speed up the process.

This solutions have been proposed some decades ago, but are still commonly used and adopted in many image processing software tools, such as Matlab, which in particular uses the rules described by Guo and Hall [3]. This is a modification of the original Zhang and Suen algorithm, to better cope with 2×2 squares and diagonal lines, similar to the one proposed by Lü and Wang [7] or Chen and Hsu [1].

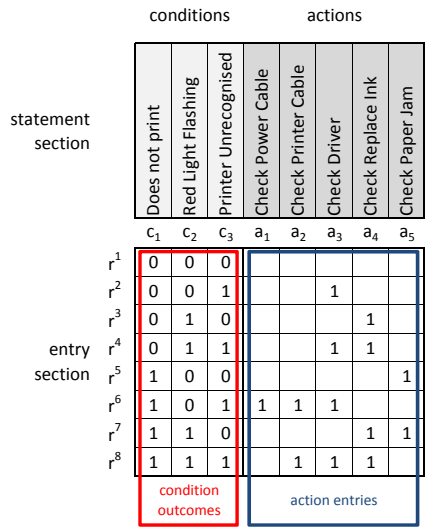

Figure 1. Decision table example.

Recently we proposed a technique for efficient neighborhood exploration using decision trees, with an example application to labeling [2]. In this work we propose to apply this approach to two thinning algorithms, namely Zhang and Suen (ZS) and Holt *et al.* (HSCP), by constructing an optimal decision tree which allows to dramatically reduce the number of memory accesses to be performed in order to explore the neighborhood. Quantitative results performed on a large freely available dataset of scanned document images will show that our solution is more effective than a look up table and that HSCP can run faster than ZS nine times out of ten.

2. Decision tables and decision trees

Local algorithms in image processing can be defined as those algorithms in which the output value for each image pixel, depends on the value of the pixel itself and of its neighbors. Accordingly, we can model local algo-

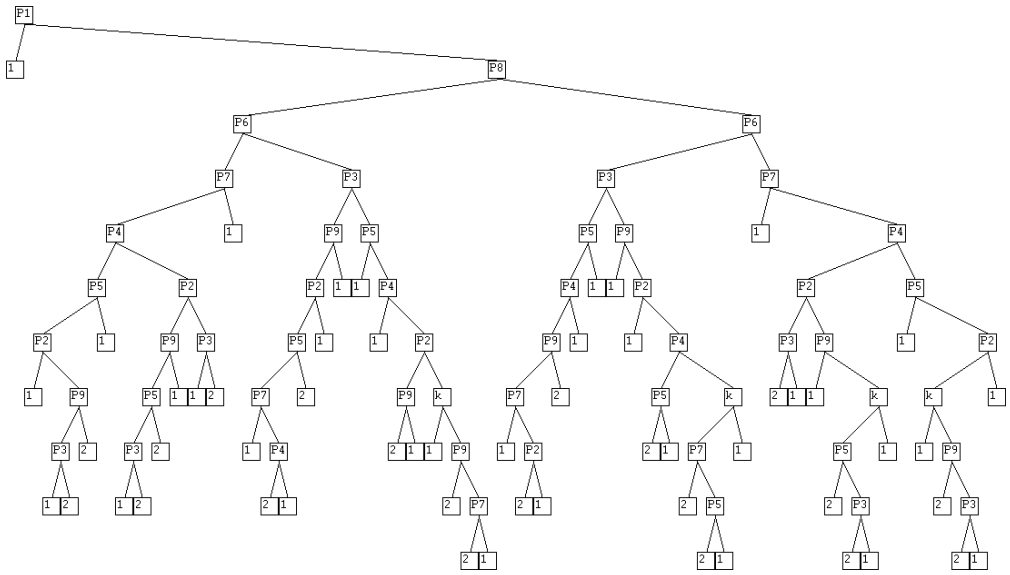

Figure 2. Decision tree for Zhang and Suen thinning algorithm

rithms by means of *decision tables*, in which the pixels values are conditions to be tested and the output is chosen by the action corresponding to the conditions outcome.

A decision table is a tabular form that presents a set of conditions and their corresponding actions. It is divided into four quadrants: an example is provided in Fig. 1. The statement section reports a set of conditions which must be tested and a list of actions to perform. Each combination of condition entries (*condition outcomes*) is paired to an *action entry*. In the action entries, a column is marked, for example with a "1", to specify whether the corresponding action is to be performed. If the conditions outcomes may only be true or false, the table is called *limited entry decision table* [9]. These will be the tables type used throughout this paper. The interpretation of a decision table is straightforward: all actions marked with 1s in the action entries vector should be performed if the corresponding outcome is obtained when testing the conditions [2].

The definition of decision tables requires all conditions to be tested in order to select the corresponding actions to be executed. Testing the conditions of the decision table has a cost which is related to the number of conditions and to the computational cost of each test. If we assume that each test has the same cost, which is true in our application, the only parameter which can be optimized is the number of conditions to be tested.

There are a number of cases in which not all conditions must be tested in order to perform the corresponding action. This observation suggests that the order with which the conditions are verified impacts on the number of tests required, thus on the total cost of testing.

We aim at finding the optimal ordering of conditions tests, which effectively produces a sequence of tests, depending on the outcome of previous tests. This is well represented by an optimal *decision tree*: the sequence requiring the minimum number of tests corresponds to the decision tree with the minimum number of nodes. The transformation of the decision table in an optimal decision tree has been deeply studied in the past and we use the Dynamic Programming technique proposed by Schumacher [10], which guarantees to obtain an optimal solution.

3. Thinning as a decision problem

Both iterative thinning algorithms considered in this paper belong to the class of parallel thinning algorithms [6]: every pixel is analyzed considering its neighborhood values in the current image, but the result is written to a different output mask, so that the procedure can be easily implemented on massively parallel architectures.

The ZS algorithm consists in a two subiterations procedure in which a foreground pixel is removed if a set of conditions is satisfied. Starting from the current pixel P_1, the neighboring pixels are enumerated in clockwise order:

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

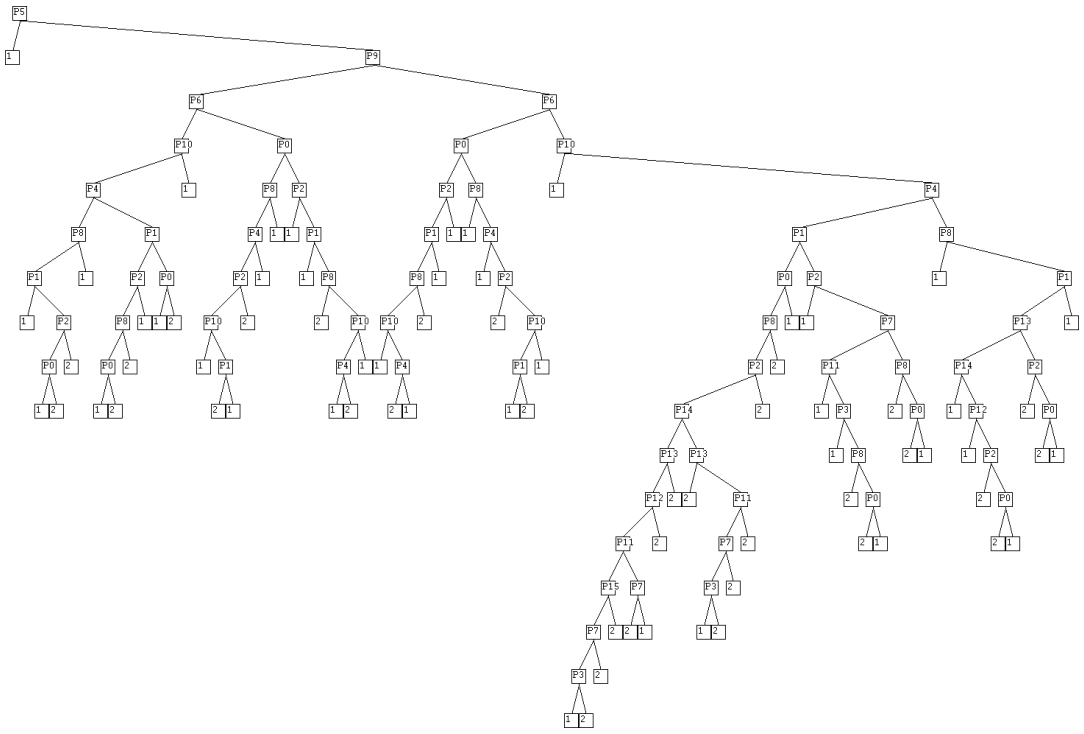

Figure 3. Decision tree for Holt et al. thinning algorithm

Let $k = 0$ during the first subiteration and $k = 1$ during the second one. Pixel P_1 should be removed if the following conditions are true:

a. $2 \leq B(P_1) \leq 6$

b. $A(P_1) = 1$

c. $P_2 * P_4 * P_6 = 0$ if $k = 0$

c'. $P_2 * P_4 * P_8 = 0$ if $k = 1$

d. $P_4 * P_6 * P_8 = 0$ if $k = 0$

d'. $P_2 * P_6 * P_8 = 0$ if $k = 1$

where $A(P_1)$ is the number of 01 patterns in clockwise order and $B(P_1)$ is the number of non zero neighbors of P_1.

The HSCP algorithm is built on the ZS algorithm by defining an *edge* function $E(P)$ which returns true if browsing the neighborhood in clockwise order there are one or more 00 patterns, one or more 11 patterns and exactly one 01 pattern ([5], Appendix A). The algorithm thus has a single subiteration which removes a foreground pixel if the following conditions are true:

1. $E(P_1) = 1$
2. $E(P_4) * P_2 * P_6 = 0$
3. $E(P_6) * P_8 * P_4 = 0$
4. $E(P_4) * E(P_5) * E(P_6) = 0$

It should be noted that the edge function requires checking all neighbors of the analyzed pixel, thus the windows used by the HSCP algorithm has size 4×4.

Chen and Hsu [1] observed that given the eight neighbors of P_1 the outcome of the conditions is known, thus they built two look-up tables (LUT) for the two subiterations and used the pixel values as bits for the index of the LUT. This allows to save all the operations required to compute $A(P_1)$, $B(P_1)$ and the other two conditions, adding only one memory access.

The LUT approach suggests that these thinning techniques can be modeled as decision tables in which the conditions are given by the fact that a neighboring pixel belongs to the foreground, and the only two possible actions are removing the current pixel or not. The ZS algorithm has also another condition, that is the value of subiteration index k. This results in a 9 conditions decision table for the ZS algorithm (512 rules) and 16 conditions (the pixels of a 4×4 window) for HSCP algorithm (65536 rules). We ran the dynamic programming technique referred in Section 2 obtaining the two optimal decision trees shown in Fig. 2 and in Fig. 3. These trees represent the best access order for the neighborhood of each pixel. The leaves of the trees are the two actions: 1 means "do nothing", while 2 means "remove". The left branch should be taken if the pixel referred in a node is background, otherwise the algorithm should follow the

Table 1. Comparison of the different thinning strategies and algorithms

	Average ms	fastest
ZS	1633	0%
ZS+LUT	1693	0%
ZS+Tree	1495	9%
HSCP	2493	0%
HSCP+Tree	1371	91%

right one. In Fig. 3, the pixels in the 4×4 neighborhood are numbered in row major ordering.

4. Results

To test the effectiveness of our proposals we compared the original ZS and HSCP with their version based on optimal decision trees. We also compared the performance of the LUT version of the ZS algorithm. The procedures were used to thin a set of binary document images, composed by 6105 high resolution scans of books taken from the Gutenberg Project [8], with an average amount of 1.3 millions of pixels. This is a typical application of document analysis and character recognition where thinning is a commonly employed preprocessing step. The tests have been performed on a Intel Core 2 Duo E6420 processor, using a single core for the processing.

The results of the comparison are reported in Table 1. The use of the decision trees significantly improves the performance of both ZS and HSCP algorithms. Moreover the surprising result is that the use of a LUT does not lead to any improvement over the straightforward implementation of ZS, probably because the added cost of composing the 8 bit index with bitwise operations and the additional memory access are comparable with the cost of the original operations. A second important result is that on average HSCP, despite being slower then ZS on sequential machines, becomes the fastest approach when the memory access is optimized with our proposal. In fact on the 91% of the cases turns out to be the fastest solution, mainly because the overall cost of an iteration is strongly reduced, thus the low number of iterations becomes the key factor in its success. With respect to the original ZS technique, the tree based version is around 10% faster, while HSCP is improved of around a 45%. This is supported by the observation that the larger the window, the higher the saving can be. HSCP+Tree is around 20% faster than the original ZS approach.

5. Conclusion

In this paper a systematic approach to minimize the number of memory accesses during neighborhood exploration has been applied to two widely employed iterative parallel thinning methodologies. Results show that significant improvement may be obtained on real world problems.

The use of decision trees saves many memory accesses, by complicating the branching structure. While this could potentially impact negatively on the performance, because of wrong branch predictions, experimental results prove that the reduced cost for memory access greatly overcomes this problem. Our interpretation is that the structure of the caches in modern architectures is still not able to completely remove the cost of accessing memory in two dimensional structured data.

The optimization of the search order can be applied to many other binary image processing operations, which require neighborhood analysis and optimization.

References

[1] Y.-S. Chen and W.-H. Hsu. A modified fast parallel algorithm for thinning digital patterns. *Pattern Recogn Lett*, 7(2):99–106, 1988.

[2] C. Grana and D. Borghesani. Optimal decision tree synthesis for efficient neighborhood computation. In *Proceedings of the XI Conference of the Italian Association for Artificial Intelligence (AIXIA09)*, pages 92–101, Reggio Emilia, Italy, Dec. 2009.

[3] Z. Guo and R. W. Hall. Parallel thinning with two-subiteration algorithms. *Commun ACM*, 32(3):359–373, 1989.

[4] R. W. Hall. Fast Parallel Thinning Algorithms: Parallel Speed and Connectivity Preservation. *Commun ACM*, 32(1):124–131, 1989.

[5] C. M. Holt, A. Stewart, M. Clint, and R. H. Perrott. An Improved Parallel Thinning Algorithm. *Commun ACM*, 30(2):156–160, 1987.

[6] L. Lam, S.-W. Lee, and C. Y. Suen. Thinning Methodologies—A Comprehensive Survey. *IEEE T Pattern Anal*, 14(9):869–885, 1992.

[7] H. E. Lü and P. S. P. Wang. A Comment on "A Fast Parallel Algorithm for Thinning Digital Patterns". *Commun ACM*, 29(3):239–242, 1986.

[8] Project Gutenberg. http://www.gutenberg.org.

[9] L. T. Reinwald and R. M. Soland. Conversion of Limited-Entry Decision Tables to Optimal Computer Programs I: Minimum Average Processing Time. *J ACM*, 13(3):339–358, 1966.

[10] H. Schumacher and K. C. Sevcik. The Synthetic Approach to Decision Table Conversion. *Commun ACM*, 19(6):343–351, 1976.

[11] T. Y. Zhang and C. Y. Suen. A Fast Parallel Algorithm for Thinning Digital Patterns. *Commun ACM*, 27(3):236–239, 1984.