Distanceless Label Propagation: an Efficient Direct Connected Component Labeling Algorithm for GPUs

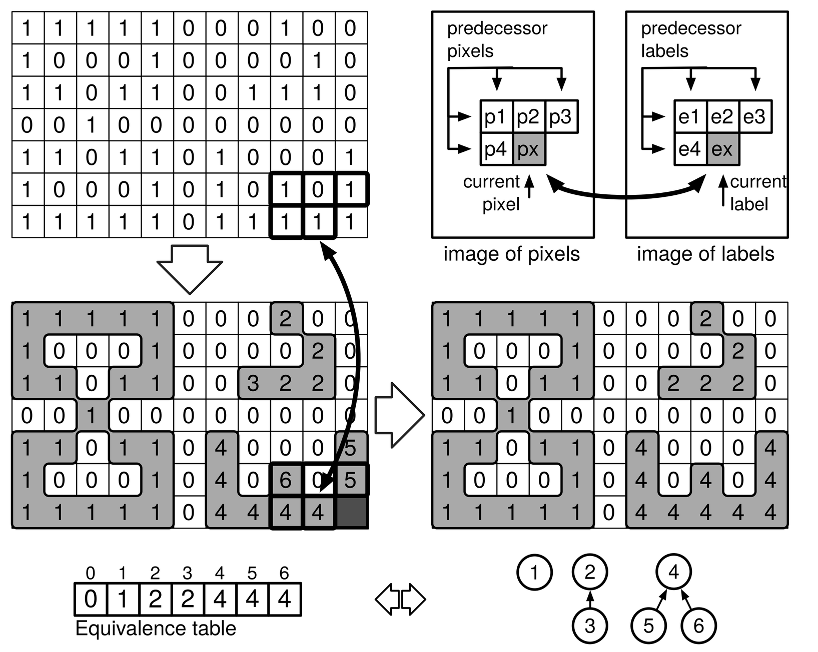
Laurent Cabaret, Lionel Lacassagne, Daniel Etiemble, MICS CentraleSupélec, LIP6 UPMC, LRI Univ. Paris-Sud

Context

Most Connected Component Labeling (CCL) algorithms are sequential, direct and optimized for CPU. Very few were designed specifically for GPU architecture. The most efficient GPU implementations are iterative in order to manage synchronizations between processing units but the number of iterations depends on the image shape and density. DLP-GPU is a GPU dedicated direct algorithm.

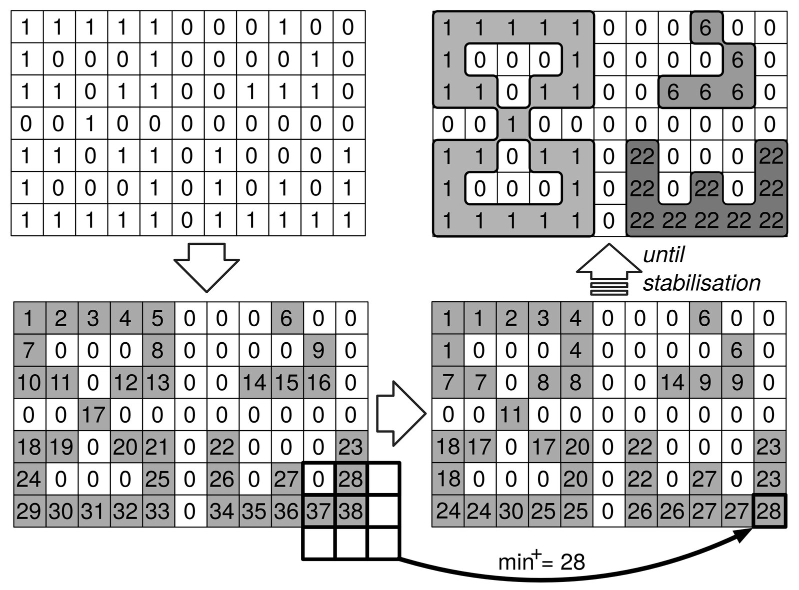
Iterative vs Direct Algorithms

Direct (CPU)



Classical direct algorithms process the input image pixel by pixel with a neighborhood mask and an equivalence table that holds a graph structure (oriented forest) to represent the label connections. This linear scanning is not suitable for GPU implementation.

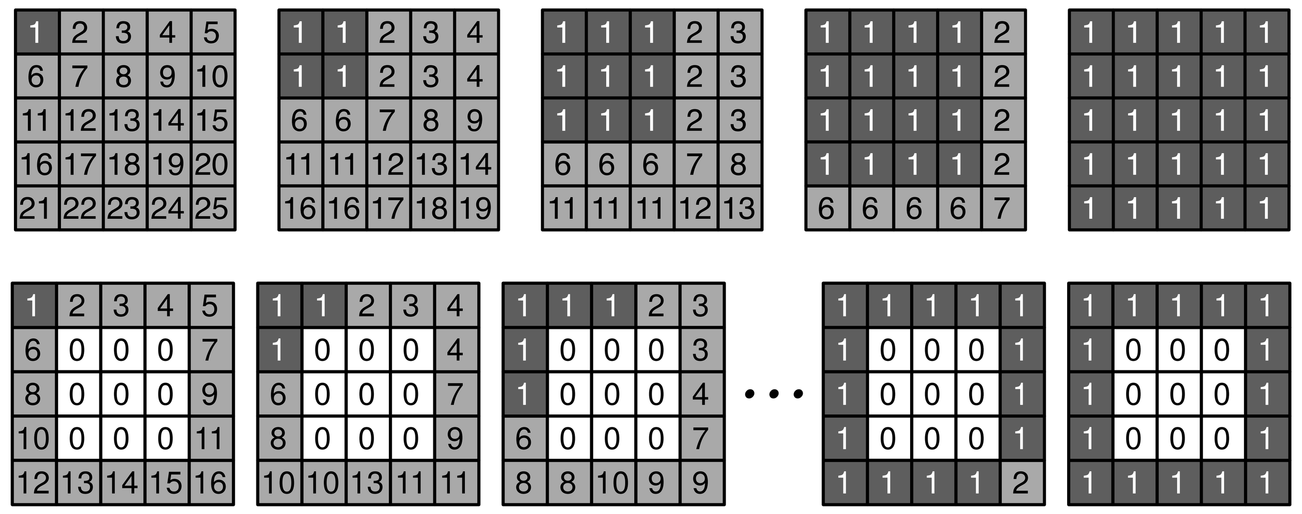
Iterative



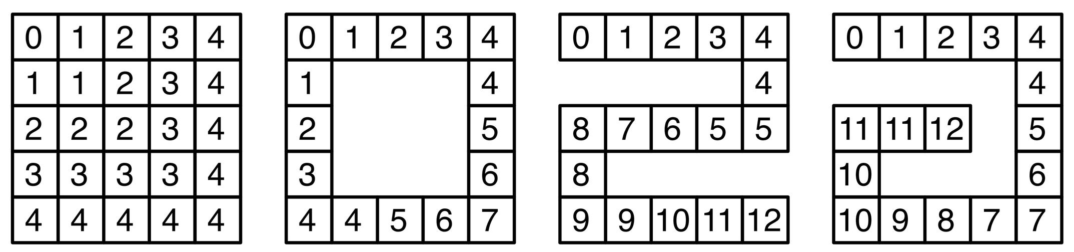
Iterative algorithms propagate localy (mask's horizon) the connection between pixels and iterate until stabilization. While this process helps from the synchronization point of view, it introduces a strong dependency to the image shape and density increasing the whole labeling time.

Iterative propagation, shape dependancy, geodesic distance

While a full square can be labeled in 5 iterations a square with a hole needs 8 iterations.



In one iteration, the propagation distance of a label is limited to 1 by the mask radius. The number of iterations is $(gd+1)$ with gd the geodesic distance. gd is data-dependant and reflect the image complexity. For the spiral (one of the worst cases) the number of iterations dramaticaly increases with the spiral size.



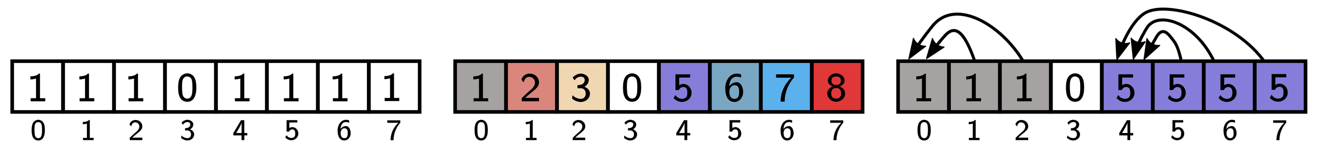
Spiral width	Iterations
5	13
100	5001
2048	2.1×10^6

DLP: beyond the mask's horizon

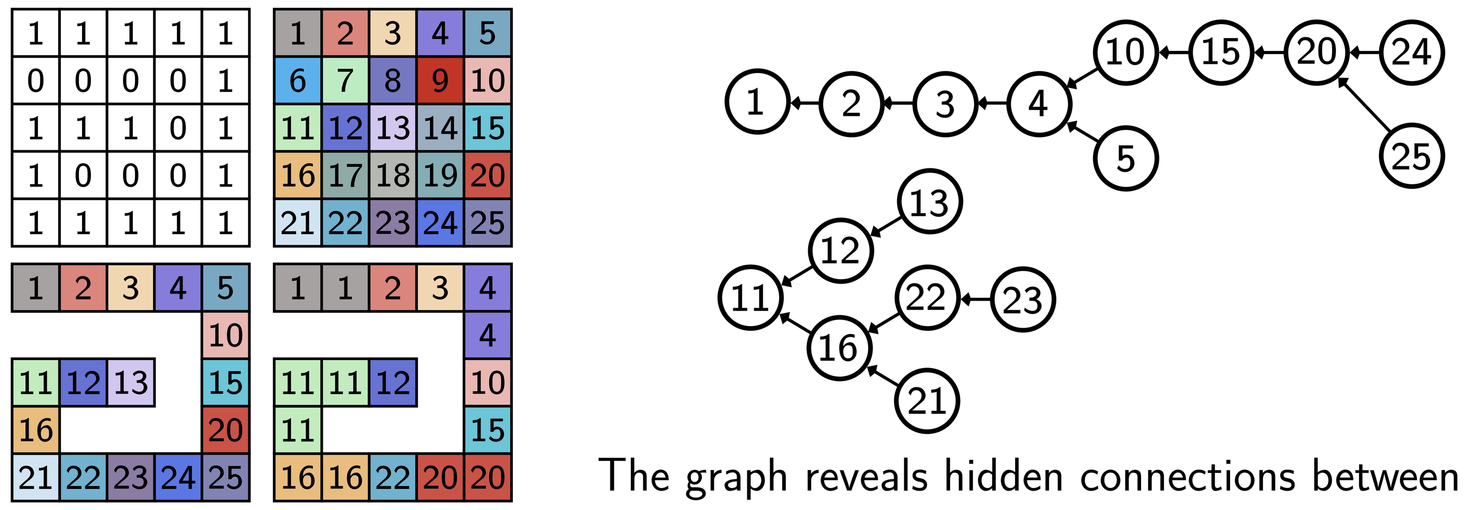
DLP-I: labels initialization → embed the graph

By initializing the label image with a value refering to each pixel position, a label points to its original location.

1D:



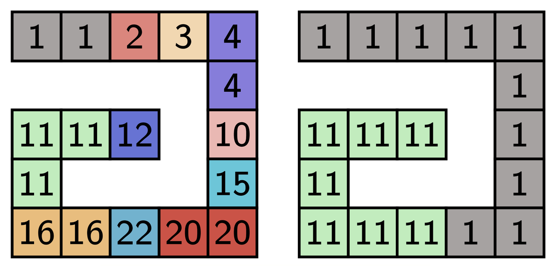
2D:



The graph reveals hidden connections between labels

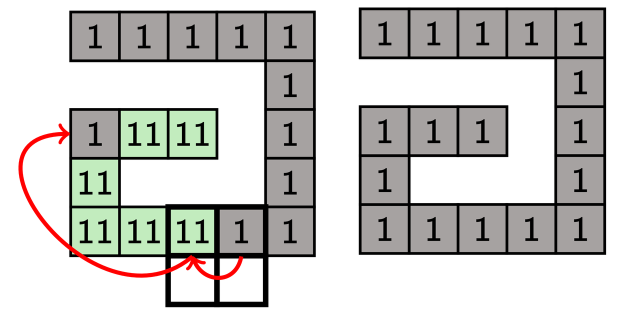
DLP-R: Relabeling

By setting each pixel site to the value of its corresponding root, the relabeling of the image and the transitive closure of the graph are performed simultaneously



DLP-SetRoot: adding Union-Find

DLP-SetRoot analyse the label values in the neighborhood to find the minimum (ε) and then assigns it to all the neighbor roots (e_k).



The spiral (regardless its size) can be solved using once the sequence: {DLP-I, DLP-SetRoot, DLP-R, DLP-SetRoot, DLP-R}. This sequence has to be compared to the 2.1×10^6 iterations required by classical iterative algorithms.

DLP-RUF: Recursive Union-Find

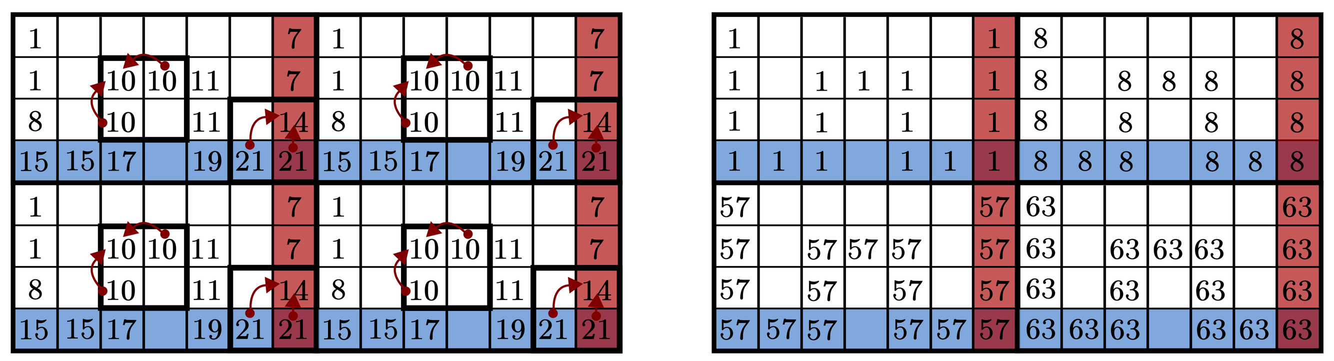
On GPU, several threads can modify the graph concurrently. DLP-RUF addresses the concurrency issue with a recursive call to *atomicRUF* (based on *atomicMin*). Although DLP-RUF can label the whole image in one pass, its efficiency is increased after an optional {DLP-SetRoot, DLP-R sequence}.

Algorithm 1: $atomicRUF(E, e_k, \varepsilon)$

```
if e_k > ε then
    minResult = atomicMin(&E[e_k − 1], ε)
if ε > minResult then
    ▹ minResult < ε < e_k
    atomicRUF(E, ε, minResult)
else
    if e_k > minResult then
        ▹ ε < minResult < e_k
        atomicRUF(E, minResult, ε)
```

DLP-GPU: fitted to GPU architecture

The image is sliced into tiles to take advantage of the shared memory. Each tile is locally labelled using DLP mechanisms with a 2×2 propagation mask. Local labels are translated to global labels before applying the one pass border merging (east and south) with DLP-RUF. Then a final global relabeling with DLP-R is applied.



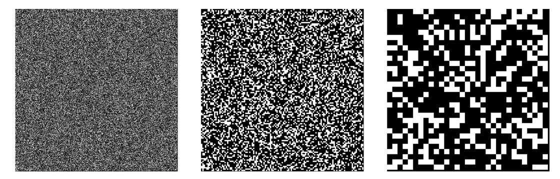
DLP-GPU structure

Step 1: tile local labeling [in shared memory]
- DLP-I*(tile)*
- DLP-SetRoot*(tile)* [optional]
- DLP-R*(tile)* [optional]
- DLP-RUF*(tile)*
- DLP-R*(tile)*
- Label translation

Step 2: border merging [in global memory]

Step 3: whole image relabeling [in global memory]

Reproducible benchmarks

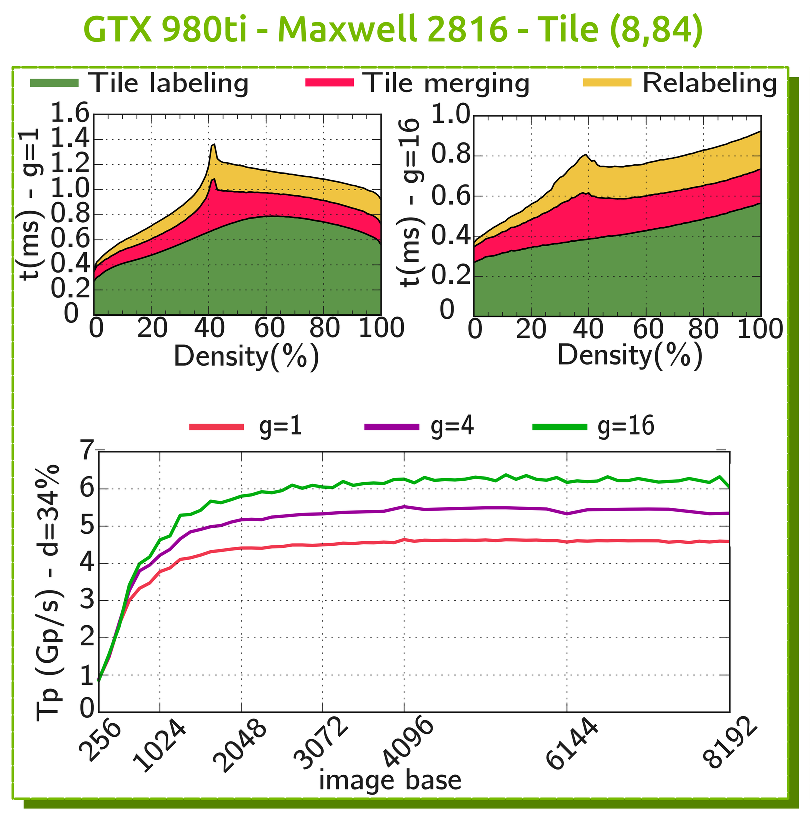


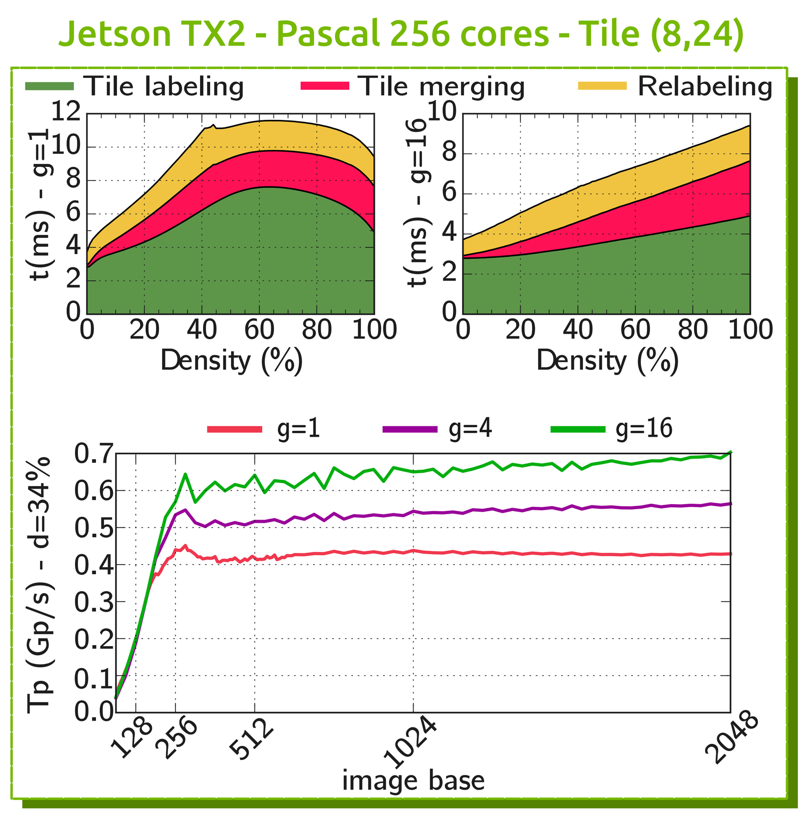
Random-based but reproducible benchmarks (Mersene Twister + fixed seed) allow fine analysis and fair comparisons between algorithms.

The image's parameters are: the density d (between 0 and 100%) the granularity g (between 1 and 16) and the image's size.

Results - fixed (2048×2048) and variable size

With only 256 cores vs 2816 (9.1%), the Jetson TX2 achieve 11.2% of the GTX980ti performance. For both cards, when g increases, the curves come closer to a straight line. The border processing (step 2) is very cycle consuming : quite the same time than relabeling (step 3) while it only processes borders (noncoalescent accesses to the east borders pixels and to the equivalence table within the image).





The throughput performance increases with g and the peak performance is quickly reached. The image size for which half of the peak performance ($N_{1/2}$ metric) is reached is respectively 640 × 640 and 176 × 176. DLP-GPU quickly reaches the peak performance of the GPU.

Conclusion

DLP is a new direct CCL algorithm for GPU. Thanks to a recursive union-find with atomic instructions, DLP is no more iterative but direct like the algorithms for multi-core processors. The equivalence table is embedded within the image in order to reduce the number of memory accesses and also to simplify and combine transitive closure and relabeling operations. The optionnal pre-processing step can speeds up DLP-GPU from ×1.5 for low density images up to ×2.5 for high density images.

CentraleSupélec · MICS · LIP6 · UPMC Sorbonne Universités · Université Paris Sud