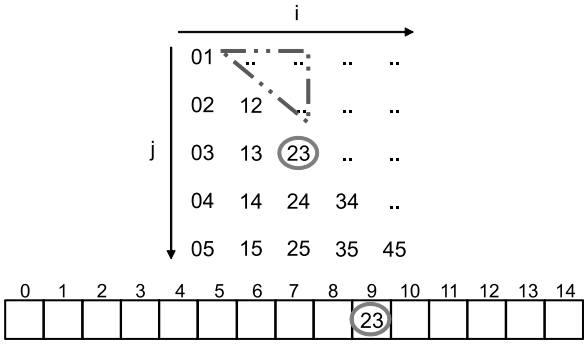

Fig. 1. An illustration of the two-to-one transformation.

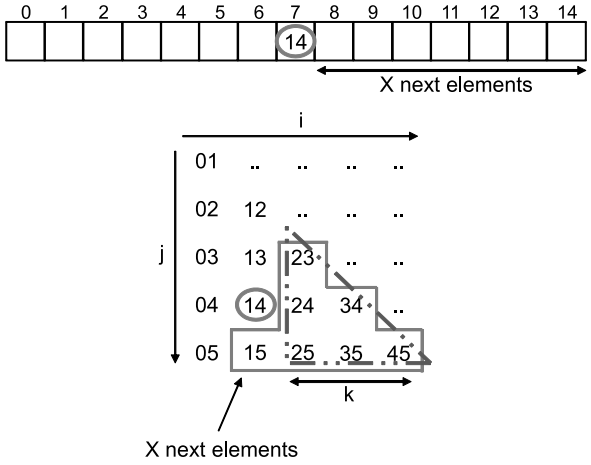

Fig. 2. An illustration of the one-to-two transformation.

APPENDIX A
MAPPING PROOFS

A.1 Two-to-one Index Transformation

Let us consider a 2D abstraction in which the elements of the neighborhood are disposed in a zero-based indexing 2D representation. This repartition is performed in a similar way as a lower triangular matrix. Let n be the size of the solution representation and let $m = \frac{n \times (n-1)}{2}$ be the size of its neighborhood. Let i and j be the indexes of two elements to be exchanged in a permutation. A candidate neighbor is then identified by both i and j indexes in the 2D abstraction. Let $f(i,j)$ be the corresponding index in the 1D neighborhood fitnesses structure. Fig. 1 is an example illustrating this abstraction.

In this example, $n = 6, m = 15$ and the neighbor identified by the coordinates ($i = 2$, $j = 3$) is mapped to the corresponding 1D array element $f(i,j) = 9$.

The neighbor represented by the (i,j) coordinates is known and its corresponding index $f(i,j)$ on the 1D structure has to be calculated. If the 1D array size was $n * n$, the 2D abstraction would be similar to a matrix and thus the mapping would be:

$$f(i,j) = i \times (n-1) + (j-1)$$

Since the 1D array size is $m = \frac{n \times (n-1)}{2}$, in the 2D abstraction, elements above the diagonal preceding the neighbor do not have to be considered (illustrated in Fig. 1 by a triangle). The corresponding two-to-one transformation is therefore:

$$f(i,j) = i \times (n-1) + (j-1) - \frac{i \times (i+1)}{2} \quad (1)$$

A.2 One-to-two Index Transformation

Let us consider the 2D abstraction previously presented. If the element corresponding to $f(i,j)$ in the 2D abstraction has a given i abscissa, then let k be the distance plus one between the $i+1$ and $n-2$ abscissas. If k is known, the value of i can be deduced:

$$i = n - 2 - \lfloor \frac{\sqrt{8X+1} - 1}{2} \rfloor \quad (2)$$

Let X be the number of elements following $f(i,j)$ in the neighborhood index-based array numbering:

$$X = m - f(i,j) - 1 \quad (3)$$

Since this number can be also represented in the 2D abstraction, the main idea is to maximize the distance k such as:

$$\frac{k \times (k+1)}{2} \leq X \quad (4)$$

Fig. 2 gives an illustration of this idea (represented by a triangle).

Resolving (4) yields the greatest distance k:

$$k = \lfloor \frac{\sqrt{8X+1} - 1}{2} \rfloor \quad (5)$$

A value of i can then be calculated according to (2). Finally, by using (1), j can be given by:

$$j = f(i,j) - i \times (n-1) + \frac{i \times (i+1)}{2} + 1 \quad (6)$$

A.3 3-exchange Neighborhood

An instance of a large neighborhood is a neighborhood built by exchanging three values. Variants of this neighborhood such as 3-opt have been used for permutation problems [1].

For an array of size n, the size of this neighborhood is $\frac{n \times (n-1) \times (n-2)}{6}$. A mapping here between a neighbor and a GPU thread is also particularly challenging. One-to-three and three-to-one index transformations must be handled efficiently.

The mapping for this neighborhood is a generalization of the 2-Hamming distance neighborhood with a third index (see A.4 and A.5). The complexity of the mappings is logarithmic in practice i.e. it depends on the numerical Newton-Raphson method (solving cubic equation).

A.4 One-to-three Index Transformation

$f(x, y, z)$ is a given index of the 1D neighborhood fitnesses structure and the objective is to find the three indexes x, y and z. Let n be the size of the solution representation and $m = \frac{n \times (n-1) \times (n-2)}{6}$ be the size of the neighborhood. The main idea is to find in which plan (coordinate z) corresponds the given element $f(x, y, z)$ in the 3D abstraction. If this corresponding plan is found, then the rest is similar to the one-to-two index transformation. Figure 3 illustrates an example of the 3D abstraction.

In this representation, since each plan is a 2D abstraction, the number of elements in each plan is the number of combinations C_k^2 where $k \in \{2, 3, \ldots, n-1\}$ according to each plan. For a specific neighbor, if a value of k is found, then the value of the corresponding plan z is:

$$z = n - k - 1 \tag{7}$$

For a given index $f(x, y, z)$ belonging to the plan k in the 3D abstraction, the number of elements contained in the next plans is C_k^2 (also equal to $\frac{k \times (k-1) \times (k-2)}{6}$).

Let Y be the number of elements following $f(x, y, z)$ in both the 1D neighborhood fitnesses structure and the 3D abstraction:

$$Y = m - f(x, y, z)$$

Then the main idea is to minimize k such as:

$$\frac{k \times (k-1) \times (k-2)}{6} >= Y \tag{8}$$

By reordering (8), in order to find a value of k, the next step is to solve the following equation:

$$k_1^3 - k_1 - 6Y = 0 \tag{9}$$

Cardano's method in theory allows to solve cubic equation. Nevertheless, in the case of finite discrete machine, this method can lose precision especially for big integers. As a consequence, a simple Newton-Raphson method for finding an approximate value of k_1 is enough for our problem. Indeed, this iterative process follows a set guideline to approximate one root, considering the function, its derivative, an initial arbitrary k_1-value and a certain precision (see Algorithm 1).

Algorithm 1 Newton-Raphson method for solving $k_1^3 - k_1 - 6Y = 0$.

```
k1 ← initial_value;
repeat
    term ← (k1 * k1 * k1 − k1 − 6 * Y) / (3 * k1 * k1 − 1);
    k1 ← k1 − term;
until |term / k1| > precision
```

Finally, since the minimization of k in (8) is expected, the value of k is:

$$k = \lceil k_1 \rceil$$

Then a value of z can be deduced with (7). At this step, the plan corresponding to the element $f(x, y, z)$ is known. The next steps for finding x and y are identically the same as the one-to-two index transformation with a change of variables.

First, the number of elements preceding $f(x, y, z)$ in the neighborhood index-based array numbering is exactly:

$$nbElementsBefore = m - \frac{(k+1) \times k \times (k-1)}{6}$$

Second, the number of elements contained in the same plan z as $f(x, y, z)$ is:

$$nbElements = \frac{k \times (k-1)}{2}$$

Finally the index of the last element of the plan z is:

$$lastElement = nbElementsBefore + nbElements - 1$$

As a result, the one-to-two index transformation is applied with a change of variables:

$$f(i, j) = f(x, y, z) - nbElementsBefore$$

$$n' = n - (z + 1)$$

$$X = lastElement - f(x, y, z)$$

After performing this transformation, a value of x and y can be deduced:

$$x = i + (z + 1)$$

$$y = j + (z + 1)$$

A.5 Three-to-one index transformation

x, y and z are known and its corresponding index $f(x, y, z)$ have to be found. According to the 3D abstraction, since a value of z is known, k can be calculated:

$$k = n - 1 - z$$

Then the number of elements preceding $f(x, y, z)$ in the neighborhood index-based array numbering can be also deduced.

If each plan size was $(n-2)*(n-2)$, each 2D abstraction would be similar to a matrix and the $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ mapping would be:

$$f_1(x, y, z) = z \times (n-2) \times (n-2) + (x-1) \times (n-2) + (y-2) \tag{10}$$

Since each 2D abstraction looks like a triangular matrix, some elements must not be considered. The advantage of the 3D abstraction is that these elements can be found by geometric construction (see Fig. 4).

First, given a plan z, the number of elements in the previous plans to not consider is:

$$n1 = z \times (n-2) \times (n-2) - nbElementsBefore$$

Second, the number of elements on the left side to not consider in the plan z is:

$$n2 = z \times (n-2)$$

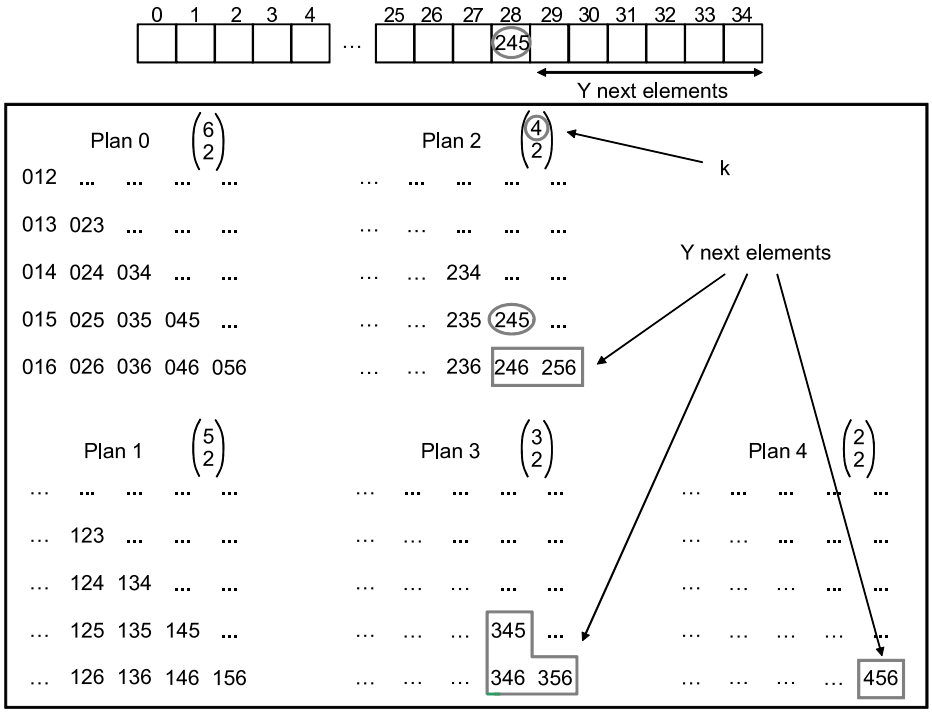

Fig. 3. An illustration of the one-to-three transformation.

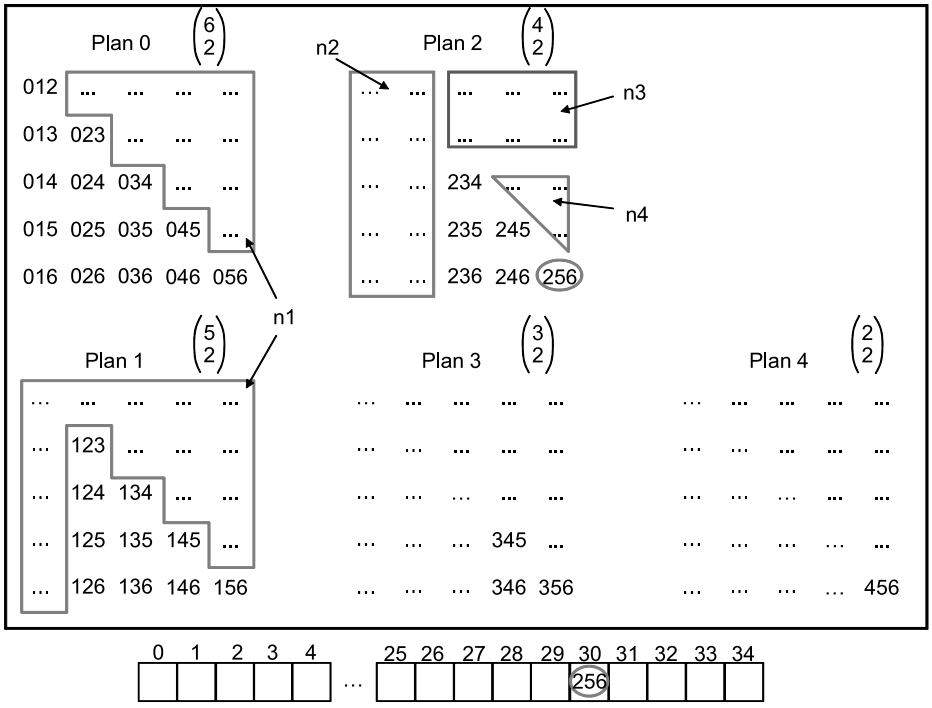

Fig. 4. $\mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ mapping.

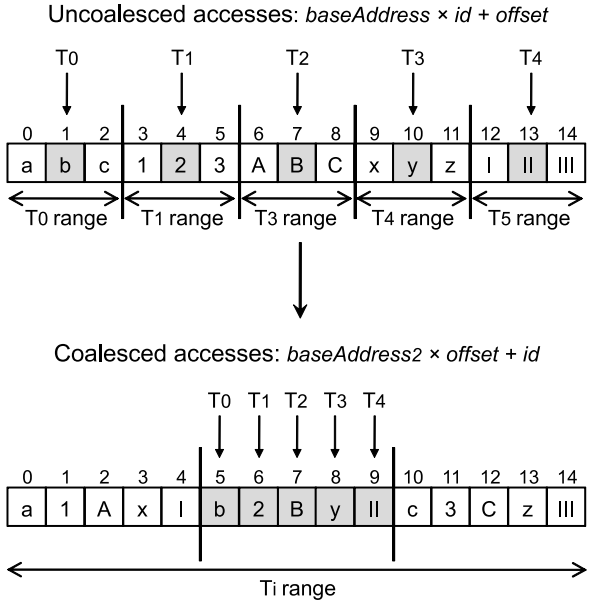

Fig. 5. An example of coalescing transformation for local structures.

Third, the number of elements on the upper side to not consider in the plan z is:

$$n3 = (y - z) \times (n - k - 1)$$

Fourth, the number of elements on the upper triangle above $f(x, y, z)$ to not consider is:

$$n4 = \frac{(y - z) \times (y - z - 1)}{2}$$

Finally a value of $f(x, y, z)$ can be deduced:

$$f(x, y, z) = f_1(x, y, z) - n1 - n2 - n3 - n4 \quad (11)$$

APPENDIX B
MEMORY MANAGEMENT OF LOCAL SEARCH METAHEURISTICS ON GPU

B.1 Coalescing Transformation

For additional local structures which are particular to a given thread, memory coalescing on global memory can be performed. Fig. 5 exhibits an example of a coalescing transformation for local structures. As illustrated in the top of the figure, a natural wrong approach to arrange the elements is to align the different structures one after the other. Thereby, each thread can access to the elements of its own structure with a natural pattern $baseAddress \times id + offset$. For instance, in the figure, each thread access to the second element of its structure with $baseAddress = 3$ and $offset = 2$.

Even if this way of organizing the elements on global memory is natural, it is clearly not efficient. Indeed, to get a better global memory performance, memory accesses must constitute a contiguous range of addresses to be coalesced. This is done in the bottom of the figure. In the second approach, the elements of the structures are dispatched such that thread accesses will be coalesced into a single memory transaction. In the figure, for instance, accessing to the second element is done by using the pattern $baseAddress2 \times offset + id$. This transformation mechanism is well-adapted for the case where each neighbor uses a large private structure which cannot be stored in local private memory. An experimental comparison of the two approaches is conducted in Section C.3.

B.2 Shared memory

Even if the shared memory has been widely investigated to reduce non-coalesced accesses in regular applications (e.g. [2], [3]), its use may not be well-adapted for the parallel iteration-level model. Due to the limited capacity of each multiprocessor (varying from 16KB to 48KB), data inputs such as matrices cannot be completely stored on shared memory. Thus, the use of shared memory must be considered as a user-managed cache. This implies an explicit effort of code transformation: one has to identify common sub-structures which are likely to be concurrently accessed by threads of a same block in accordance with the SIMD execution model. Unfortunately, such common accesses are not always predictable in evaluation functions since most of access patterns to data inputs differ from a neighbor to another (especially for permutation-based problems). A specific code transformation and its associated performance results for the shared memory are described in Section C.1.

APPENDIX C
EXPERIMENTS

C.1 Application to the Quadratic Assignment Problem

The following transformation intends to show how to take advantage of the shared memory in the case of the QAP. The Δ calculation (slight variation) of the evaluation function for a neighbor (i,j) is given by:

$$\begin{aligned}\Delta = (a_{ii} - a_{jj}) \times (b_{\pi(j)\pi(j)} - b_{\pi(i)\pi(i)}) \quad (12)\\ + (a_{ii} - a_{jj}) \times (b_{\pi(j)\pi(i)} - b_{\pi(i)\pi(j)})\end{aligned}$$

$$\begin{aligned}\Delta = \Delta + \sum_{\substack{k=0 \\ k \neq i \\ k \neq j}}^{n} (a_{ki} - a_{kj}) \times (b_{\pi(k)\pi(j)} - b_{\pi(k)\pi(i)}) \quad (13)\\ + (a_{ik} - a_{jk}) \times (b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)})\end{aligned}$$

A depth look at (13) indicates that a and b sub-matrices involving the variable k might be concurrently accessed in parallel. Therefore, the idea is to associate such sub-structures with the shared memory as a user-managed cache. The equation (13) can be transformed into another one:

$$\Delta = \Delta + \sum_{\substack{k=0 \\ k \neq i \\ k \neq j \\ ra \leftarrow row(a,k) \\ rb \leftarrow row(b,\pi(k)) \\ ca \leftarrow col(a,k) \\ cb \leftarrow col(b,\pi(k))}}^{n} (ra_i - ra_j) \times (rb_{\pi(j)} - rb_{\pi(i)}) + (ca_i - ca_j) \times (ca_{\pi(j)} - cb_{\pi(i)}) \quad (14)$$

Rows and columns are copied on shared memory at the beginning of each loop iteration via a synchronization mechanism. In this manner, accesses to these structures are performed through this memory in (14). Therefore, sub-matrices can benefit from the shared memory since it is a low-latency memory in which non-coalescing accesses are reduced. However, this transformation is problem-dependent and it might not be applied to some evaluation functions.

Table 17 reports the obtained results for an implementation based on shared memory. In a general manner, the shared memory version ($\mathrm{GPU_{Sh}}$) obtains better performance results than the basic GPU version without memory optimization. The acceleration factors in comparison with a single-core on CPU diversify between $\times 0.7$ and $\times 16.1$ for $\mathrm{GPU_{Sh}}$ against $\times 0.5$ and $\times 15.7S$ for the standard version.

Nevertheless, such an improvement does not occur in regards with the texture version. Indeed, for the two first configurations, the shared memory version is clearly outperformed by the texture one. This is certainly due to the extra cost of data copies from global memory to shared memory (thus extra non-coalesced accesses) including local synchronizations for each loop iteration. In the third configuration, the gap is less important since global memory is easier to access due to the relaxation of the coalescing rules.

A conclusion from this experiment indicates that the use of shared memory gives an additional performance improvement. However, on the one hand, an effort of code rewriting has to be provided. Furthermore, it is not clear that such a transformation is always feasible. On the other hand, performance results for the QAP indicate that this version is dominated by the texture one especially for low-graphic cards configurations. Therefore, it seems that the stand-alone use of shared memory is not well-adapted for the parallel iteration-level.

C.2 Application to the Permuted Perceptron Problem

As previously said, the definition of the neighborhood is a major step in the performance of the algorithm. Indeed, theoretical and experimental studies have shown that the increase of the neighborhood size may improve the quality of the obtained solutions [4]. Nevertheless, as it is generally CPU time-consuming, this mechanism is not often fully exploited in practice. To deal with such an issue, only the use of massive parallelism allows to design methods based on large neighborhood structures. The next experiment intends to point out an instance of such a very large neighborhood for a Hamming distance of three for the PPP. Experimental results are reported in Table 2. In a general way, for the same instance, the obtained acceleration factors are much more important than for the previous neighborhoods. For example, for the texture version, the obtained speed-ups already vary from $\times 3.2$ to $\times 45.8$ just for the first four PPP instances.

The conclusion from this experiment indicates that GPUs are efficient to deal with very large neighborhoods. Indeed, such a neighborhood for the PPP was unpractical in terms of single CPU computational resources. Hence, implementing this algorithm on the GPU allows to exploit this data-parallelism. Even if the CPU execution was too much time consuming from larger instances than $m = 201$ and $n = 217$, we are convinced that GPU computing could be exploited for designing large algorithms to improve the quality of solutions.

C.3 Coalescing accesses to global memory

The evaluation function of a neighbor for the PPP requires the calculation of a structure called histogram. Since this structure is particular to a neighbor, the local memory is managed to store the histogram. However, for large instances (from $m = 401$ and $n = 417$), the amount of local memory may be not enough to store this structure and the program will fail at compilation time. As a consequence, in that case, the histogram must be stored on global memory.

As previously said, coalescing on global memory is a must to obtain the global best performance. In Section B.1, two different access patterns have been described to deal with large local structures on global memory. Even if the first one is natural, this associated performance might be limited because of non-coalesced memory accesses. That is the reason why, the second pattern has been used for the previous experiments. To confirm this point, a next experiment consists in comparing the performance results obtained by the two different approaches (Table 29).

In comparison with the second approach, the obtained performance results are drastically reduced. Indeed, for the standard GPU version, the speed-up obtained from the first approach varies between $\times 4.6$ and $\times 7.0$. It is about five times less than the second approach. The same phenomenon occurs for the texture version (accelerations from $\times 6.3$ to $\times 7.9$) whilst the speed-up for the second approach alternates from $\times 28.8$ and $\times 44.1$. A conclusion of this experiment indicates that memory coalescing applied on local structures is a must to obtain the best performance.

C.4 Application to the Weierstrass Continuous Function

As previously seen, the obtained accelerations factors on GPU vary accordingly to the instance size of the contin-

TABLE 1
Measures in terms of efficiency of a shared memory version. The quadratic assignment problem using a pair-wise-exchange neighborhood is considered.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores			Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		
	GPU	GPU_{Tex}	GPU_{Sh}	GPU	GPU_{Tex}	GPU_{Sh}
tai30a	$4.2_{\times 0.5}$	$1.3_{\times \mathbf{1.8}}$	$3.1_{\times 0.7}$	$2.3_{\times 0.8}$	$1.0_{\times \mathbf{1.9}}$	$2.1_{\times 0.9}$
tai35a	$6.5_{\times 0.5}$	$1.6_{\times \mathbf{2.1}}$	$5.0_{\times 0.7}$	$2.9_{\times 0.9}$	$1.2_{\times \mathbf{2.3}}$	$2.8_{\times 1.0}$
tai40a	$9.7_{\times 0.5}$	$1.8_{\times \mathbf{2.6}}$	$5.5_{\times 0.9}$	$3.7_{\times \mathbf{1.1}}$	$1.5_{\times \mathbf{2.9}}$	$3.5_{\times \mathbf{1.2}}$
tai50a	$16_{\times 0.6}$	$3.0_{\times \mathbf{3.2}}$	$8.9_{\times \mathbf{1.1}}$	$5.7_{\times \mathbf{1.4}}$	$1.8_{\times \mathbf{4.6}}$	$5.0_{\times \mathbf{1.7}}$
tai60a	$28_{\times 0.6}$	$4.9_{\times \mathbf{3.4}}$	$13_{\times \mathbf{1.3}}$	$8.4_{\times \mathbf{1.6}}$	$2.0_{\times \mathbf{7.1}}$	$6.1_{\times \mathbf{2.3}}$
tai80a	$63_{\times 0.7}$	$10_{\times \mathbf{4.2}}$	$41_{\times \mathbf{1.5}}$	$19_{\times \mathbf{1.9}}$	$4.5_{\times \mathbf{8.1}}$	$9.6_{\times \mathbf{3.7}}$
tai100a	$139_{\times 0.8}$	$19_{\times \mathbf{5.6}}$	$67_{\times \mathbf{1.6}}$	$33_{\times \mathbf{2.3}}$	$8.7_{\times \mathbf{8.8}}$	$15.6_{\times \mathbf{4.8}}$
Instance	**Intel Xeon E5450 GeForce GTX 280 240 GPU cores**			**Xeon E5620 Tesla M2050 448 GPU cores**		
	GPU	GPU_{Tex}	GPU_{Sh}	GPU	GPU_{Tex}	GPU_{Sh}
tai30a	$1.1_{\times \mathbf{1.4}}$	$0.8_{\times \mathbf{2.0}}$	$1.0_{\times \mathbf{1.7}}$	$0.5_{\times \mathbf{2.2}}$	$0.4_{\times \mathbf{2.8}}$	$0.5_{\times \mathbf{2.4}}$
tai35a	$1.2_{\times \mathbf{1.9}}$	$0.9_{\times \mathbf{2.6}}$	$1.1_{\times \mathbf{2.2}}$	$0.6_{\times \mathbf{2.4}}$	$0.5_{\times \mathbf{3.8}}$	$0.6_{\times \mathbf{2.7}}$
tai40a	$1.3_{\times \mathbf{2.7}}$	$1.1_{\times \mathbf{3.3}}$	$1.2_{\times \mathbf{3.0}}$	$0.7_{\times \mathbf{3.2}}$	$0.5_{\times \mathbf{4.4}}$	$0.6_{\times \mathbf{3.8}}$
tai50a	$1.7_{\times \mathbf{4.1}}$	$1.3_{\times \mathbf{5.3}}$	$1.5_{\times \mathbf{4.7}}$	$0.8_{\times \mathbf{5.4}}$	$0.6_{\times \mathbf{7.2}}$	$0.7_{\times \mathbf{6.1}}$
tai60a	$2.0_{\times \mathbf{5.6}}$	$1.6_{\times \mathbf{7.3}}$	$1.7_{\times \mathbf{6.5}}$	$1.1_{\times \mathbf{8.4}}$	$0.9_{\times \mathbf{10.2}}$	$1.0_{\times \mathbf{9.1}}$
tai80a	$3.2_{\times \mathbf{9.1}}$	$2.8_{\times \mathbf{10.8}}$	$2.9_{\times \mathbf{10.3}}$	$1.9_{\times \mathbf{12.4}}$	$1.7_{\times \mathbf{13.5}}$	$1.8_{\times \mathbf{12.7}}$
tai100a	$5.5_{\times \mathbf{10.9}}$	$3.7_{\times \mathbf{16.5}}$	$4.1_{\times \mathbf{14.6}}$	$3.1_{\times \mathbf{15.7}}$	$2.6_{\times \mathbf{18.6}}$	$3.0_{\times \mathbf{16.1}}$

TABLE 2
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of three (binary representation).

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}
73-73	$742_{\times 0.7}$	$167_{\times \mathbf{3.2}}$	$11.1_{\times \mathbf{28.1}}$	$10.6_{\times \mathbf{29.4}}$	$9.8_{\times \mathbf{28.4}}$	$9.2_{\times \mathbf{30.2}}$	$8.1_{\times \mathbf{34.3}}$	$7.9_{\times \mathbf{35.1}}$
81-81	$1023_{\times 0.8}$	$242_{\times \mathbf{3.3}}$	$16_{\times \mathbf{29.5}}$	$15_{\times \mathbf{31.4}}$	$13.3_{\times \mathbf{31.2}}$	$12.9_{\times \mathbf{32.2}}$	$11.2_{\times \mathbf{37.0}}$	$10.8_{\times \mathbf{38.4}}$
101-117	$3329_{\times 0.9}$	$883_{\times \mathbf{3.6}}$	$57_{\times \mathbf{32.1}}$	$55_{\times \mathbf{33.3}}$	$50_{\times \mathbf{35.6}}$	$47_{\times \mathbf{37.9}}$	$43_{\times \mathbf{41.4}}$	$41_{\times \mathbf{43.4}}$
201-217	$70325_{\times 1.1}$	$15471_{\times \mathbf{5.1}}$	$650_{\times \mathbf{39.5}}$	$615_{\times \mathbf{41.8}}$	$510_{\times \mathbf{43.9}}$	$488_{\times \mathbf{45.8}}$	$411_{\times \mathbf{54.5}}$	$398_{\times \mathbf{56.3}}$

TABLE 3
Measures of the benefits of using coalescing accesses to the global memory on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered.

Instance	CPU	Non-coalesced accesses		Coalesced accesses	
		GPU	GPU_{Tex}	GPU	GPU_{Tex}
401-417	403	$88_{\times \mathbf{4.6}}$	$64_{\times \mathbf{6.3}}$	$20_{\times \mathbf{20.1}}$	$14_{\times \mathbf{28.8}}$
601-617	2049	$328_{\times \mathbf{6.2}}$	$249_{\times \mathbf{8.2}}$	$67_{\times \mathbf{30.5}}$	$51_{\times \mathbf{40.1}}$
801-817	5410	$801_{\times \mathbf{6.8}}$	$665_{\times \mathbf{8.1}}$	$154_{\times \mathbf{35.3}}$	$128_{\times \mathbf{42.3}}$
1001-1017	11075	$1577_{\times \mathbf{7.0}}$	$1361_{\times \mathbf{8.1}}$	$292_{\times \mathbf{37.9}}$	$252_{\times \mathbf{43.9}}$
1301-1317	25016	$3582_{\times \mathbf{7.0}}$	$3180_{\times \mathbf{7.9}}$	$651_{\times \mathbf{38.5}}$	$568_{\times \mathbf{44.1}}$

uous Weierstrass function. Another experiment consists in seeing the impact of varying the neighborhood size in terms of performance. Table 4 reports the produced results for different neighborhood sizes with the dimension fixed to 2. The experiments were not carried out for more than 100000 neighbors since these executions are unpractical on CPU.

Whatever the used configuration, one can clearly see that the acceleration factors grow accordingly with the neighborhood size. These speed-ups alternate from $\times 3.7$ to $\times 232.4$ according to the different configurations. For smaller neighborhoods (e.g. 100, 500 and 1000), since one neighbor is at least associated with one thread, the number of neighbors is not enough to cover the memory access latencies. This can explain this performance degradation especially for advanced cards in which the number of available multiprocessors is more important.

C.4.1 Quality of the solutions

Regarding the quality of the obtained solutions, the final results for CPU and GPU versions are different. Indeed, according to [5], GPU floating-point computations used in the Weierstrass function such as sin, sqrt or pow introduce a unit of least precision error. An analysis of the accuracy of the solutions is detailed in Table 38 for the third configuration. The average and the standard deviation

TABLE 4
Measures in terms of efficiency for the Weierstrass function using single precision. The problem of dimension 2 is considered for different neighborhood sizes.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
100	179	$17_{\times \mathbf{10.5}}$	57	$11_{\times \mathbf{5.2}}$	41	$11_{\times \mathbf{3.7}}$	38	$6.4_{\times \mathbf{5.9}}$
500	482	$22_{\times \mathbf{21.9}}$	287	$15_{\times \mathbf{19.1}}$	203	$12_{\times \mathbf{16.9}}$	194	$6.9_{\times \mathbf{28.1}}$
1000	1152	$31_{\times \mathbf{37.1}}$	576	$16_{\times \mathbf{36.0}}$	407	$13_{\times \mathbf{31.3}}$	399	$7.3_{\times \mathbf{54.7}}$
5000	5317	$137_{\times \mathbf{38.1}}$	2874	$32_{\times \mathbf{89.8}}$	2034	$20_{\times \mathbf{101.7}}$	2018	$12_{\times \mathbf{168.1}}$
10000	10298	$247_{\times \mathbf{41.7}}$	5752	$52_{\times \mathbf{110.6}}$	4088	$32_{\times \mathbf{127.8}}$	4051	$18_{\times \mathbf{215.1}}$
20000	21330	$428_{\times \mathbf{49.8}}$	11488	$101_{\times \mathbf{113.7}}$	8508	$65_{\times \mathbf{130.9}}$	8490	$38_{\times \mathbf{223.4}}$
50000	53402	$1060_{\times \mathbf{50.3}}$	28718	$244_{\times \mathbf{117.7}}$	20364	$148_{\times \mathbf{137.6}}$	20291	$89_{\times \mathbf{227.6}}$
100000	104439	$2124_{\times \mathbf{51.6}}$	58857	$493_{\times \mathbf{119.4}}$	40700	$292_{\times \mathbf{139.4}}$	40213	$173_{\times \mathbf{232.4}}$

are reported for the number of iterations representing the algorithm divergence and convergence. The maximal error of the fitnesses produced on GPU has been calculated in comparison with the CPU version during the first iteration of the algorithm. This information explains why the different versions take a different execution path during the first iterations (divergence). Indeed, since 10000 neighbors are considered with a small radius, the algorithm is quite sensitive to such a precision error. The use of double precision allows to obtain a better accuracy leading to delay this divergence. However, the algorithm still takes rapidly a divergent branch. In addition to this, the performance is drastically reduced by a factor of four (speed-ups varying from $\times 31.8$ to $\times 39.6$) when using the double precision. Regarding the convergence of the different versions for a pre-determined threshold, it clearly differs from a version to another. Statistical tests cannot conclude if the distribution of the averages is different i.e. if a version is better than another.

Regarding the final results, even if they are different, the produced solutions for GPU versions are valid from an optimization point of view. Indeed, a look at the final solution vector does not exhibit any incoherent values. Furthermore, after re-calculating on CPU the best fitness found on GPU, the precision error generally varies from 10^{-3} to 10^{-5}. Therefore, regarding optimization problems, such a precision error does not seem to be a critical issue.

C.5 Thread Control

The thread control prevents the application from crashing and may enhance the LSM performance rather than a parameters tuning made at compile time. The next experiment intends to highlight the benefits of applying thread control in terms of performance optimization. The considered problem is the PPP using a neighborhood based on a hamming distance of two. Table 6 reports the obtained results. In a general manner, performance results obtained for the thread control ($\mathrm{GPU}_{\mathrm{TexTC}}$) are significantly improved in comparison with its counterpart without control. Indeed, the accelerations factors alternate from $\times 3.8$ to $\times 81.4$ for $\mathrm{GPU}_{\mathrm{TexTC}}$ against $\times 3.6$ to $\times 73.3$. This is quite significant since the results correspond to a performance enhancement from 5% to 20%. In comparison with the TSP, such an improvement can be explained by the fact that the neighborhood is smaller. That the reason why, the dynamic heuristic spent less time to find a good parameters auto-tuning.

C.6 Analysis of the Data Transfer

As previously said, generating the neighborhood on CPU and evaluating on GPU may lead to a significant amount of data transferred. So, the generation of the neighborhood on GPU is likely to deal with such an issue. Table 7 reports the time spent by each operation in the two approaches by using a neighborhood based on a Hamming distance of one (n neighbors). The third configuration was used for this experiment. For the first approach, one can notice that the time spent by the data transfer is significant. It represents almost 25% of the total execution time for each instance. In the second approach, in comparison with the previous one, the time spent on the data transfer is drastically reduced with the instance size increase. Indeed, for the instance $m = 73$ and $n = 73$, this time corresponds to 19% of the total running time and it reaches the value of 1% for the last instance ($m = 1301$ and $n = 1317$). Another observation concerns the time taken by the generation and the evaluation of the neighborhood on GPU (evaluation kernel). Generally speaking, the algorithm in the second approach takes advantage of resource use since most of the total running time is dedicated to the GPU kernel execution. For example, in the fourth instance $m = 201$ and $n = 217$, the time associated with the evaluation of the neighborhood accounts for 86% of the total execution time. This time grows along with the instance size (more than 90% for the other larger instances).

As a result, the second approach outperforms the first one in terms of efficiency. Indeed, regarding the related acceleration factors for the two approaches, the reported results are in accordance with the previous observations. This difference of performance tends to grow with the instance size.

TABLE 5
Analysis of the solutions accuracy for the Weierstrass function using single and double precision on the GTX 280 (vector of real values).

Instance	CPU	conv.	GPU	max error	div.	conv.	GPU_{double}	max error	div.	conv.
1	2034	$8203_{\pm 2312}$	$16_{\times \mathbf{127.1}}$	1.8×10^{-2}	$3_{\pm 2}$	$8634_{\pm 2932}$	$62_{\times \mathbf{31.8}}$	1.1×10^{-4}	$15_{\pm 4}$	$8021_{\pm 2233}$
2	4088	$6729_{\pm 2811}$	$32_{\times \mathbf{127.8}}$	2.0×10^{-2}	$5_{\pm 3}$	$7122_{\pm 3917}$	$125_{\times \mathbf{32.7}}$	1.9×10^{-4}	$13_{\pm 3}$	$7434_{\pm 2745}$
3	6113	$7421_{\pm 2537}$	$47_{\times \mathbf{130.0}}$	1.4×10^{-2}	$4_{\pm 2}$	$7634_{\pm 2335}$	$179_{\times \mathbf{34.1}}$	2.4×10^{-4}	$14_{\pm 3}$	$8584_{\pm 2932}$
4	8137	$6776_{\pm 3402}$	$61_{\times \mathbf{133.4}}$	7.1×10^{-2}	$3_{\pm 2}$	$8647_{\pm 2936}$	$220_{\times \mathbf{37.0}}$	1.5×10^{-4}	$13_{\pm 4}$	$5907_{\pm 3138}$
5	10225	$8504_{\pm 2043}$	$76_{\times \mathbf{134.5}}$	5.7×10^{-2}	$3_{\pm 2}$	$8612_{\pm 2318}$	$271_{\times \mathbf{37.7}}$	1.7×10^{-4}	$13_{\pm 4}$	$6935_{\pm 3982}$
6	12193	$8852_{\pm 1907}$	$90_{\times \mathbf{135.5}}$	1.9×10^{-2}	$6_{\pm 4}$	$7639_{\pm 3972}$	$318_{\times \mathbf{38.2}}$	1.1×10^{-4}	$19_{\pm 6}$	$8894_{\pm 2736}$
7	14319	$7530_{\pm 2572}$	$104_{\times \mathbf{137.7}}$	3.1×10^{-2}	$7_{\pm 5}$	$8014_{\pm 1722}$	$369_{\times \mathbf{38.8}}$	1.2×10^{-4}	$17_{\pm 5}$	$7201_{\pm 2480}$
8	16699	$6364_{\pm 2145}$	$120_{\times \mathbf{139.2}}$	9.4×10^{-2}	$5_{\pm 3}$	$7430_{\pm 2411}$	$429_{\times \mathbf{38.9}}$	1.4×10^{-4}	$18_{\pm 5}$	$8537_{\pm 1968}$
9	19008	$7624_{\pm 3413}$	$134_{\times \mathbf{141.9}}$	4.7×10^{-2}	$4_{\pm 2}$	$8194_{\pm 3720}$	$482_{\times \mathbf{39.4}}$	1.3×10^{-4}	$13_{\pm 4}$	$7259_{\pm 3905}$
10	21095	$8301_{\pm 2741}$	$148_{\times \mathbf{142.5}}$	8.4×10^{-2}	$5_{\pm 2}$	$8627_{\pm 2889}$	$533_{\times \mathbf{39.6}}$	1.8×10^{-4}	$15_{\pm 5}$	$8635_{\pm 2954}$

TABLE 6
Measures of the benefits of applying thread control. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered.

Instance	Core 2 Duo T5800 GeForce 8600M GT		Core 2 Quad Q6600 GeForce 8800 GTX		Xeon E5450 GeForce GTX 280		Xeon E5620 Tesla M2050	
	GPU_{Tex}	GPU_{TexTC}	GPU_{Tex}	GPU_{TexTC}	GPU_{Tex}	GPU_{TexTC}	GPU_{Tex}	GPU_{TexTC}
73-73	$0.8_{\times \mathbf{3.6}}$	$0.8_{\times \mathbf{3.8}}$	$0.2_{\times \mathbf{10.1}}$	$0.2_{\times \mathbf{10.3}}$	$0.2_{\times \mathbf{10.9}}$	$0.2_{\times \mathbf{11.6}}$	$0.2_{\times \mathbf{12.3}}$	$0.2_{\times \mathbf{12.8}}$
81-81	$1.1_{\times \mathbf{3.8}}$	$1.0_{\times \mathbf{4.2}}$	$0.3_{\times \mathbf{10.4}}$	$0.3_{\times \mathbf{10.9}}$	$0.2_{\times \mathbf{12.2}}$	$0.2_{\times \mathbf{12.8}}$	$0.2_{\times \mathbf{13.4}}$	$0.2_{\times \mathbf{13.7}}$
101-117	$2.5_{\times \mathbf{4.4}}$	$2.4_{\times \mathbf{4.6}}$	$0.6_{\times \mathbf{12.4}}$	$0.6_{\times \mathbf{13.7}}$	$0.4_{\times \mathbf{18.1}}$	$0.4_{\times \mathbf{19.9}}$	$0.3_{\times \mathbf{22.0}}$	$0.3_{\times \mathbf{23.2}}$
201-217	$15_{\times \mathbf{4.7}}$	$13_{\times \mathbf{5.4}}$	$3.3_{\times \mathbf{15.4}}$	$2.9_{\times \mathbf{17.6}}$	$1.9_{\times \mathbf{25.3}}$	$1.6_{\times \mathbf{30.1}}$	$1.6_{\times \mathbf{30.6}}$	$1.4_{\times \mathbf{34.9}}$
401-417	$103_{\times \mathbf{5.4}}$	$92_{\times \mathbf{6.2}}$	$24_{\times \mathbf{18.3}}$	$21_{\times \mathbf{21.0}}$	$14_{\times \mathbf{28.8}}$	$13_{\times \mathbf{31.2}}$	$10_{\times \mathbf{38.3}}$	$8.9_{\times \mathbf{43.0}}$
601-617	$512_{\times \mathbf{6.3}}$	$446_{\times \mathbf{7.2}}$	$89_{\times \mathbf{28.3}}$	$78_{\times \mathbf{32.3}}$	$51_{\times \mathbf{40.1}}$	$45_{\times \mathbf{45.3}}$	$35_{\times \mathbf{58.4}}$	$31_{\times \mathbf{65.9}}$
801-817	$1245_{\times \mathbf{6.9}}$	$1064_{\times \mathbf{8.1}}$	$212_{\times \mathbf{32.8}}$	$182_{\times \mathbf{38.1}}$	$128_{\times \mathbf{42.3}}$	$109_{\times \mathbf{49.6}}$	$81_{\times \mathbf{67.1}}$	$72_{\times \mathbf{75.5}}$
1001-1017	$2421_{\times \mathbf{7.2}}$	$2087_{\times \mathbf{8.4}}$	$409_{\times \mathbf{35.2}}$	$350_{\times \mathbf{41.3}}$	$252_{\times \mathbf{43.9}}$	$216_{\times \mathbf{51.3}}$	$154_{\times \mathbf{71.9}}$	$138_{\times \mathbf{80.2}}$
1301-1317	$4903_{\times \mathbf{8.0}}$	$4265_{\times \mathbf{9.2}}$	$911_{\times \mathbf{36.2}}$	$779_{\times \mathbf{42.5}}$	$568_{\times \mathbf{44.1}}$	$485_{\times \mathbf{51.7}}$	$342_{\times \mathbf{73.3}}$	$308_{\times \mathbf{81.4}}$

TABLE 7
Measures of the benefits of generating the neighborhood on GPU. on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of one is considered.

Instance	CPU	Evaluation on GPU				Generation and evaluation on GPU			
		GPU_{Tex}	process	transfers	kernel	GPU_{Tex}	process	transfers	kernel
73-73	1.1	$3.4_{\times 0.3}$	4.0%	22.7%	73.3%	$3.0_{\times 0.4}$	1.0%	23.2%	75.8%
81-81	1.3	$3.8_{\times 0.3}$	5.3%	25.9%	68.8%	$3.3_{\times 0.4}$	1.1%	22.9%	75.9%
101-117	2.2	$5.1_{\times 0.4}$	4.8%	24.5%	70.7%	$4.2_{\times 0.5}$	1.1%	19.1%	79.8%
201-217	8.1	$11_{\times 0.7}$	6.8%	25.3%	67.9%	$7.7_{\times \mathbf{1.1}}$	1.2%	12.0%	86.8%
401-417	31	$27_{\times \mathbf{1.2}}$	7.1%	25.0%	67.9%	$14_{\times \mathbf{2.2}}$	1.1%	6.2%	92.7%
601-617	105	$68_{\times \mathbf{1.5}}$	7.1%	26.1%	66.8%	$43_{\times \mathbf{2.4}}$	1.0%	3.9%	95.1%
801-817	200	$98_{\times \mathbf{2.0}}$	7.1%	24.2%	68.7%	$50_{\times \mathbf{4.0}}$	0.6%	1.4%	98.0%
1001-1017	336	$106_{\times \mathbf{3.2}}$	5.5%	23.5%	71.0%	$58_{\times \mathbf{5.8}}$	0.3%	0.6%	99.1%
1301-1317	687	$146_{\times \mathbf{4.7}}$	5.2%	22.4%	72.4%	$85_{\times \mathbf{8.0}}$	0.2%	0.4%	99.4%

C.7 Additional Data Transfer Optimization

In our approach, data transfer from GPU to CPU corresponds to the entire fitnesses structure. In Hill Climbing, the reduction operator can help to reduce these copying operations. Table 8 highlights the analysis of the time dedicated for each major operation for a neighborhood based on a Hamming distance of two. On the one hand, for the second approach, whatever the size of the neighborhood used or the instance size, the data transfer is nearly constant (varying between 0.01% and 1.46%). It can be clarified by the fact that only one solution is transferred from the GPU to the CPU at each iteration. On the other hand, one can also notice that the time spent on the search process on CPU is also minimized for the second approach. Indeed, by definition, the reduction operation consists in finding the minimum which is performed on the GPU-side in a logarithmic time. While for the first approach, most of the CPU search process time corresponds to the search of the minimum in the fitnesses structure (linear time). Therefore, both minimization of the data transfers and complexity reduction can explain such an improvement of performance.

C.8 Comparison with Other Parallel and Distributed Architectures

C.8.1 Configurations

The different machines used for the experiments for COWs and grid are described in Table 9. Most of them

TABLE 8
Analysis of the time dedicated for each operation for 100 hill climbing algorithms. The permuted perceptron problem using a Hamming distance of two and the reduction operation are considered.

Instance	GPU_{Tex}			$GPUR_{Tex}$		
	process	transfers	kernel	process	transfers	kernel
73-73	19.0%	11.2%	69.8%	1.43%	1.46%	97.11%
81-81	18.8%	10.7%	70.5%	0.91%	0.98%	98.01%
101-117	18.7%	10.1%	71.2%	0.46%	0.44%	99.10%
201-217	18.5%	7.3%	74.2%	0.36%	0.11%	99.53%
401-417	18.2%	6.3%	75.5%	0.08%	0.04%	99.88%
601-617	17.7%	4.5%	77.8%	0.04%	0.02%	99.94%
801-817	13.3%	2.5%	84.2%	0.03%	0.02%	99.96%
1001-1017	12.7%	1.5%	85.8%	0.02%	0.01%	99.97%
1301-1317	10.9%	1.5%	87.6%	0.01%	0.01%	99.98%

are octo-core workstations. The different computers have been chosen accordingly to the different GPU configurations i.e. in agreement with their computational power. Such a metric has been deduced from the potential gflops delivered by the different machines.

C.8.2 COW

The analysis of the time spent is exactly in accordance with the previous results (see Table 10). The percentage dedicated to the transfer operations varies accordingly with the speed-ups observed.

Furthermore, increasing the number of machines thus the number of communications has a negative impact on the performance for small instances such as $m = 73$ and $n = 73$. Indeed, the associated time dedicated to the transfers clearly dominates the algorithm (93% and 98% for the second and the third configurations). Such a behaviour does not occur as well in the first configuration since communication is based only on an inter-core communication.

C.8.3 Grid organization

An analysis of the time dedicated to the transfers including communication time is given in Table 11 for the grid. In comparison with COWs, the transfer time corresponding to the partitions sending and the synchronization is significantly more important whatever the instance size. This can be explained by the distribution of computers among the different sites (respectively two, five and seven according to the configuration). Indeed, in COWs, such an extra inter-sites communication does not occur since the computers are directly linked by a gigabit ethernet.

APPENDIX D
STATISTICAL TESTS

Statistical analysis must be performed to ensure that the conclusions deduced from the experiments are meaningful. Furthermore, an objective is also to prove that a particular algorithm outperforms another one. However, the comparison between two average values might be not enough. Indeed, it may differ from the comparison between two distributions. Therefore, a test has to be performed to ensure the statistical significances of the obtained results. In other words, one has to determine whether an observation is likely to be due to a sampling error or not.

The first test consists in checking if the data set is normally distributed from a number of experiments above 30. This is done by applying a Kolmogorov-Smirnov's test which is a powerful and accurate method.

To compare two different distributions (i.e. whether an algorithm is better than another or not), the Student's t-test is widely used to compare averages of normal data. The prerequisites for such a test are to check the data normality (Kolmogorov-Smirnov) then to check the variances equality of the two samples. This latter can be done by the Levene's test which is an inferential statistic used to assess the equality of variances in different sample.

The statistical confidence level is fixed to 95% and the p-values are represented for all the statistical analysis tables.

TABLE 9
Parallel and distributed machines used for the experiments on COWs and Grid'5000.

Architecture	Configuration 1 Machines	gflops	Configuration 2 Machines	gflops
GPU	Core 2 Duo T5800 GeForce 8600M GT	76.8	Core 2 Quad Q6600 GeForce 8800 GTX	384
COWs	Intel Xeon E5440 8 CPU cores	90.656	4 Intel Xeon E5440 32 CPU cores	362.624
Grid	2 Intel Xeon E5440 2 × 4 CPU cores	90.656	Amd Opteron 2218 Intel Xeon E5520 2 Intel Xeon E5420 Intel Xeon E5440 40 CPU cores	406.368

Architecture	Configuration 3 Machines	gflops	Configuration 4 Machines	gflops
GPU	Intel Xeon E5450 GeForce GTX 280	981.12	Intel Xeon E5620 Tesla M2050	1106.08
COWs	11 Intel Xeon E5440 88 CPU cores	995.236	13 Intel Xeon E5440 104 CPU cores	1176.188
Grid	2 Intel Xeon E5520 2 AMD Opteron 2218 2 Intel Xeon E5520 4 Intel Xeon E5520 Intel Xeon X5570 Intel Xeon E5520 96 CPU cores	979.104	4 Intel Xeon E5520 2 AMD Opteron 2218 2 Intel Xeon E5520 4 Intel Xeon E5520 Intel Xeon X5570 Intel Xeon E5520 112 CPU cores	1160.056

TABLE 10
Analysis of the time dedicated for each operation for a cluster of workstations. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. The fourth configuration is not represented since it is very similar to the third one.

Instance	Intel Xeon E5440 8 CPU cores			4 Intel Xeon E5440 32 CPU cores			11 Intel Xeon E5440 88 CPU cores		
	process	transfers	workers	process	transfers	workers	process	transfers	workers
73-73	4.2%	45.7%	50.1%	1.8%	93.0%	5.2%	0.7%	98.8%	0.5%
81-81	3.3%	45.3%	51.4%	2.3%	91.9%	5.8%	0.7%	98.7%	0.6%
101-117	2.6%	43.1%	54.3%	3.9%	83.8%	12.3%	1.2%	97.4%	1.4%
201-217	1.9%	42.4%	55.7%	5.7%	67.8%	26.5%	4.6%	88.2%	7.2%
401-417	1.8%	39.6%	58.6%	6.5%	57.1%	36.4%	12.1%	63.8%	24.1%
601-617	0.9%	52.0%	47.1%	3.5%	64.9%	31.6%	7.8%	71.7%	20.5%
801-817	0.6%	56.5%	42.9%	2.3%	67.4%	30.3%	5.3%	75.4%	19.3%
1001-1017	0.5%	59.4%	40.1%	1.9%	70.7%	27.4%	4.0%	79.9%	16.1%
1301-1317	0.4%	62.5%	37.1%	1.6%	71.3%	27.1%	3.5%	80.6%	15.9%

TABLE 11
Analysis of the time dedicated for each operation for workstations distributed in a grid organization. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. The fourth configuration is not represented since it is very similar to the third one.

Instance	2 Intel Xeon QC E5440 8 CPU cores			Configuration 5 machines 40 CPU cores			Configuration 12 machines 96 CPU cores		
	process	transfers	workers	process	transfers	workers	process	transfers	workers
73-73	3.4%	59.5%	37.1%	2.0%	94.1%	3.9%	0.5%	99.1%	0.4%
81-81	2.6%	54.5%	42.9%	2.0%	93.5%	4.5%	0.6%	98.9%	0.5%
101-117	2.1%	53.6%	44.3%	3.1%	87.2%	9.7%	0.8%	98.2%	1.0%
201-217	1.6%	52.7%	45.7%	4.5%	74.5%	21.0%	3.5%	91.2%	5.3%
401-417	1.5%	49.9%	48.6%	5.2%	65.8%	29.0%	10.4%	70.5%	19.1%
601-617	0.8%	59.2%	40.0%	2.8%	72.0%	25.2%	6.4%	77.1%	16.5%
801-817	0.5%	63.8%	35.7%	1.8%	74.1%	24.1%	4.1%	80.2%	15.7%
1001-1017	0.4%	66.7%	32.9%	1.5%	76.6%	21.9%	3.0%	83.3%	13.7%
1301-1317	0.3%	68.3%	31.4%	1.3%	77.1%	21.6%	2.6%	86.4%	11.0%

TABLE 12
Measures in terms of efficiency for the quadratic assignment problem using a pair-wise-exchange neighborhood (permutation representation). Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores			Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores			Intel Xeon E5450 GeForce GTX 280 240 GPU cores			Xeon E5620 Tesla M2050 448 GPU cores		
	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}
tai30a	+	+	+	+	+	+	+	+	+	+	+	+
tai35a	+	+	+	+	+	+	+	+	+	+	+	+
tai40a	+	+	+	+	+	+	+	+	+	+	+	+
tai50a	+	+	+	+	+	+	+	+	+	+	+	+
tai60a	+	+	+	+	+	+	+	+	+	+	+	+
tai80a	+	+	+	+	+	+	+	+	+	+	+	+
tai100a	+	+	+	+	+	+	+	+	+	+	+	+

TABLE 13
Measures in terms of efficiency for the quadratic assignment problem using a pair-wise-exchange neighborhood (permutation representation). Test of the null hypothesis of variances equality with the Levene's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}
tai30a	+	+	+	+	+	+	+	+
tai35a	+	+	+	+	+	+	+	+
tai40a	+	+	+	+	+	+	+	+
tai50a	+	+	+	+	+	+	+	+
tai60a	+	+	+	+	+	+	+	+
tai80a	+	+	+	+	+	+	+	+
tai100a	+	+	+	+	+	+	+	+

TABLE 14
Measures in terms of efficiency for the quadratic assignment problem using a pair-wise-exchange neighborhood (permutation representation). Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}
tai30a	-	-	-	-	-	-	-	-
tai35a	-	-	-	-	-	-	-	-
tai40a	-	-	-	-	-	-	-	-
tai50a	-	-	-	-	-	-	-	-
tai60a	-	-	-	-	-	-	-	-
tai80a	-	-	-	-	-	-	-	-
tai100a	-	-	-	-	-	-	-	-

TABLE 15
Measures in terms of efficiency of a shared memory version. The quadratic assignment problem using a pair-wise-exchange neighborhood is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores			Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		
	GPU	GPU_{Tex}	GPU_{Sh}	GPU	GPU_{Tex}	GPU_{Sh}
tai30a	+	+	+	+	+	+
tai35a	+	+	+	+	+	+
tai40a	+	+	+	+	+	+
tai50a	+	+	+	+	+	+
tai60a	+	+	+	+	+	+
tai80a	+	+	+	+	+	+
tai100a	+	+	+	+	+	+
Instance	**Intel Xeon E5450 GeForce GTX 280 240 GPU cores**			**Xeon E5620 Tesla M2050 448 GPU cores**		
	GPU	GPU_{Tex}	GPU_{Sh}	GPU	GPU_{Tex}	GPU_{Sh}
tai30a	+	+	+	+	+	+
tai35a	+	+	+	+	+	+
tai40a	+	+	+	+	+	+
tai50a	+	+	+	+	+	+
tai60a	+	+	+	+	+	+
tai80a	+	+	+	+	+	+
tai100a	+	+	+	+	+	+

TABLE 16
Measures in terms of efficiency of a shared memory version. The quadratic assignment problem using a pair-wise-exchange neighborhood is considered. Test of the null hypothesis of variances equality with the Levene's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}
	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}
tai30a	+	+	+	+	+	+	+	+
tai35a	+	+	+	+	+	+	+	+
tai40a	+	+	+	+	+	+	+	+
tai50a	+	+	+	+	+	+	+	+
tai60a	+	+	+	+	+	+	+	+
tai80a	+	+	+	+	+	+	+	+
tai100a	+	+	+	+	+	+	+	+

TABLE 17
Measures in terms of efficiency of a shared memory version. The quadratic assignment problem using a pair-wise-exchange neighborhood is considered. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}
	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}	GPU_{Sh}
tai30a	-	-	-	-	-	-	-	-
tai35a	-	-	-	-	-	-	-	-
tai40a	-	-	-	-	-	-	-	-
tai50a	-	-	-	-	-	-	-	-
tai60a	-	-	-	-	-	-	-	-
tai80a	-	-	-	-	-	-	-	-
tai100a	-	-	-	-	-	-	-	-

TABLE 18
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of one (binary representation). Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores			Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores			Intel Xeon E5450 GeForce GTX 280 240 GPU cores			Xeon E5620 Tesla M2050 448 GPU cores		
	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}
73-73	+	+	+	+	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+	+	+	+	+

TABLE 19
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of one (binary representation). Test of the null hypothesis of variances equality with the Levene's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}
73-73	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+

TABLE 20
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of one (binary representation). Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}
73-73	-	-	-	-	-	-	-	-
81-81	-	-	-	-	-	-	-	-
101-117	-	-	-	-	-	-	-	-
201-217	-	-	-	-	-	-	-	-
401-417	-	-	-	-	-	-	-	-
601-617	-	-	-	-	-	-	-	-
801-817	-	-	-	-	-	-	-	-
1001-1017	-	-	-	-	-	-	-	-
1301-1317	-	-	-	-	-	-	-	-

TABLE 21
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of two (binary representation). Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores			Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores			Intel Xeon E5450 GeForce GTX 280 240 GPU cores			Xeon E5620 Tesla M2050 448 GPU cores		
	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}
73-73	+	+	+	+	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+	+	+	+	+

TABLE 22
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of two (binary representation). Test of the null hypothesis of variances equality with the Levene's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}
73-73	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+

TABLE 23
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of two (binary representation). Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}
73-73	-	-	-	-	-	-	-	-
81-81	-	-	-	-	-	-	-	-
101-117	-	-	-	-	-	-	-	-
201-217	-	-	-	-	-	-	-	-
401-417	**0.083**	-	-	-	-	-	-	-
601-617	-	-	-	-	-	-	-	-
801-817	-	-	-	-	-	-	-	-
1001-1017	-	-	-	-	-	-	-	-
1301-1317	-	-	-	-	-	-	-	-

TABLE 24
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of three (binary representation). Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores**			**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores**			**Intel Xeon E5450 GeForce GTX 280 240 GPU cores**			**Xeon E5620 Tesla M2050 448 GPU cores**		
	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}
73-73	+	+	+	+	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+	+	+	+	+

TABLE 25
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of three (binary representation). Test of the null hypothesis of variances equality with the Levene's test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores**		**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores**		**Intel Xeon E5450 GeForce GTX 280 240 GPU cores**		**Xeon E5620 Tesla M2050 448 GPU cores**	
	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}
73-73	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+

TABLE 26
Measures in terms of efficiency for the permuted perceptron problem using a neighborhood based on a Hamming distance of three (binary representation). Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores**		**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores**		**Intel Xeon E5450 GeForce GTX 280 240 GPU cores**		**Xeon E5620 Tesla M2050 448 GPU cores**	
	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}	CPU GPU	GPU GPU_{Tex}
73-73	-	-	-	-	-	-	-	-
81-81	-	-	-	-	-	-	-	-
101-117	-	-	-	-	-	-	-	-
201-217	-	-	-	-	-	-	-	-

TABLE 27
Measures of the benefits of using coalescing accesses to the global memory on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	CPU	**Non-coalesced accesses**		**Coalesced accesses**	
		GPU	GPU_{Tex}	GPU	GPU_{Tex}
401-417	+	+	+	+	+
601-617	+	+	+	+	+
801-817	+	+	+	+	+
1001-1017	+	+	+	+	+
1301-1317	+	+	+	+	+

TABLE 28
Measures of the benefits of using coalescing accesses to the global memory on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of variances equality with the Levene's test.

Instance	CPU - Nca/GPU	Nca/GPU - Ca/GPU	Nca/GPU_{Tex} - Ca/GPU_{Tex}
401-417	+	+	+
601-617	+	+	+
801-817	+	+	+
1001-1017	+	+	+
1301-1317	+	+	+

TABLE 29
Measures of the benefits of using coalescing accesses to the global memory on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	CPU - Nca/GPU	Nca/GPU - Ca/GPU	Nca/GPU_{Tex} - Ca/GPU_{Tex}
401-417	-	-	-
601-617	-	-	-
801-817	-	-	-
1001-1017	-	-	-
1301-1317	-	-	-

TABLE 30
Measures in terms of efficiency for the Weierstrass function using single precision (vector of real values with 10000 neighbors). Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores**		**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores**		**Xeon E5450 GeForce GTX 280 240 GPU cores**		**Xeon E5620 Tesla M2050 448 GPU cores**	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
1	+	+	+	+	+	+	+	+
2	+	+	+	+	+	+	+	+
3	+	+	+	+	+	+	+	+
4	+	+	+	+	+	+	+	+
5	+	+	+	+	+	+	+	+
6	+	+	+	+	+	+	+	+
7	+	+	+	+	+	+	+	+
8	+	+	+	+	+	+	+	+
9	+	+	+	+	+	+	+	+
10	+	+	+	+	+	+	+	+

TABLE 31
Measures in terms of efficiency for the Weierstrass function using single precision (vector of real values with 10000 neighbors). Test of the null hypothesis of variances equality with the Levene's test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores** CPU - GPU	**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores** CPU - GPU	**Intel Xeon E5450 GeForce GTX 280 240 GPU cores** CPU - GPU	**Xeon E5620 Tesla M2050 448 GPU cores** CPU - GPU
1	+	+	+	+
2	+	+	+	+
3	+	+	+	+
4	+	+	+	+
5	+	+	+	+
6	+	+	+	+
7	+	+	+	+
8	+	+	+	+
9	+	+	+	+
10	+	+	+	+

TABLE 32
Measures in terms of efficiency for the Weierstrass function using single precision (vector of real values with 10000 neighbors). Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores** CPU - GPU	**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores** CPU - GPU	**Intel Xeon E5450 GeForce GTX 280 240 GPU cores** CPU - GPU	**Xeon E5620 Tesla M2050 448 GPU cores** CPU - GPU
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-
5	-	-	-	-
6	-	-	-	-
7	-	-	-	-
8	-	-	-	-
9	-	-	-	-
10	-	-	-	-

TABLE 33
Measures in terms of efficiency for the Weierstrass function using single precision. The problem of dimension 2 is considered for different neighborhood sizes. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores**		**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores**		**Xeon E5450 GeForce GTX 280 240 GPU cores**		**Xeon E5620 Tesla M2050 448 GPU cores**	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
100	+	+	+	+	+	+	+	+
500	+	+	+	+	+	+	+	+
1000	+	+	+	+	+	+	+	+
5000	+	+	+	+	+	+	+	+
10000	+	+	+	+	+	+	+	+
20000	+	+	+	+	+	+	+	+
50000	+	+	+	+	+	+	+	+
100000	+	+	+	+	+	+	+	+

TABLE 34
Measures in terms of efficiency for the Weierstrass function using single precision. The problem of dimension 2 is considered for different neighborhood sizes. Test of the null hypothesis of variances equality with the Levene's test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores** CPU - GPU	**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores** CPU - GPU	**Intel Xeon E5450 GeForce GTX 280 240 GPU cores** CPU - GPU	**Xeon E5620 Tesla M2050 448 GPU cores** CPU - GPU
100	+	+	+	+
500	+	+	+	+
1000	+	+	+	+
5000	+	+	+	+
10000	+	+	+	+
20000	+	+	+	+
50000	+	+	+	+
100000	+	+	+	+

TABLE 35
Measures in terms of efficiency for the Weierstrass function using single precision. The problem of dimension 2 is considered for different neighborhood sizes. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores** CPU - GPU	**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores** CPU - GPU	**Intel Xeon E5450 GeForce GTX 280 240 GPU cores** CPU - GPU	**Xeon E5620 Tesla M2050 448 GPU cores** CPU - GPU
100	-	-	-	-
500	-	-	-	-
1000	-	-	-	-
5000	-	-	-	-
10000	-	-	-	-
20000	-	-	-	-
50000	-	-	-	-
100000	-	-	-	-

TABLE 36
Analysis of the solutions accuracy for the Weierstrass function using single and double precision on the GTX 280 (vector of real values). Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	CPU	conv.	GPU	div.	conv.	GPU_{double}	div.	conv.
1	+	+	+	+	+	+	+	+
2	+	+	+	+	+	+	+	+
3	+	+	+	+	+	+	+	+
4	+	+	+	+	+	+	+	+
5	+	+	+	+	+	+	+	+
6	+	+	+	+	+	+	+	+
7	+	+	+	+	+	+	+	+
8	+	+	+	+	+	+	+	+
9	+	+	+	+	+	+	+	+
10	+	+	+	+	+	+	+	+

TABLE 37
Analysis of the solutions accuracy for the Weierstrass function using single and double precision on the GTX 280 (vector of real values). Test of the null hypothesis of variances equality with the Levene's test.

Instance	GPU - GPU_{double} time	GPU - GPU_{double} max error	GPU - GPU_{double} div.	CPU - GPU conv.	CPU - GPU_{double} conv.
1	+	+	+	+	+
2	+	+	+	+	+
3	+	+	+	+	+
4	+	+	+	+	+
5	+	+	+	+	+
6	+	+	+	+	+
7	+	+	+	+	+
8	+	+	+	+	+
9	+	+	+	+	+
10	+	+	+	+	+

TABLE 38
Analysis of the solutions accuracy for the Weierstrass function using single and double precision on the GTX 280 (vector of real values). Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	GPU - GPU_{double} time	GPU - GPU_{double} max error	GPU - GPU_{double} div.	CPU - GPU conv.	CPU - GPU_{double} conv.
1	-	-	-	**0.147**	**0.083**
2	-	-	-	**0.072**	**0.095**
3	-	-	-	**0.085**	**0.144**
4	-	-	-	**0.128**	**0.088**
5	-	-	-	**0.143**	**0.139**
6	-	-	-	**0.095**	**0.107**
7	-	-	-	**0.073**	**0.071**
8	-	-	-	**0.147**	**0.118**
9	-	-	-	**0.107**	**0.151**
10	-	-	-	**0.142**	**0.084**

TABLE 39
Measures in terms of efficiency for the traveling salesman problem using a pair-wise-exchange neighborhood (permutation representation). Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores**			**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores**			**Intel Xeon E5450 GeForce GTX 280 240 GPU cores**			**Xeon E5620 Tesla M2050 448 GPU cores**		
	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}	CPU	GPU	GPU_{Tex}
eil101	+	+	+	+	+	+	+	+	+	+	+	+
d198	+	+	+	+	+	+	+	+	+	+	+	+
pcb442	+	+	+	+	+	+	+	+	+	+	+	+
rat783	+	+	+	+	+	+	+	+	+	+	+	+
d1291	+	+	+	+	+	+	+	+	+	+	+	+
pr2392	+	.	.	+	+	+	+	+	+	+	+	+
fnl4461	+	.	.	+	.	.	+	+	+	+	+	+
rl5915	+	.	.	+	.	.	+	.	.	+	+	+

TABLE 40
Measures in terms of efficiency for the traveling salesman problem using a pair-wise-exchange neighborhood (permutation representation). Test of the null hypothesis of variances equality with the Levene's test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores**		**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores**		**Intel Xeon E5450 GeForce GTX 280 240 GPU cores**		**Xeon E5620 Tesla M2050 448 GPU cores**	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}
eil101	+	+	+	+	+	+	+	+
d198	+	+	+	+	+	+	+	+
pcb442	+	+	+	+	+	+	+	+
rat783	+	+	+	+	+	+	+	+
d1291	+	+	+	+	+	+	+	+
pr2392	.	.	+	+	+	+	+	+
fnl4461	.	.	.	.	+	+	+	+
rl5915	.	.	.	.	.	+	+	+

TABLE 41
Measures in terms of efficiency for the traveling salesman problem using a pair-wise-exchange neighborhood (permutation representation). Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	**Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores**		**Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores**		**Intel Xeon E5450 GeForce GTX 280 240 GPU cores**		**Xeon E5620 Tesla M2050 448 GPU cores**	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}	GPU	GPU_{Tex}
eil101	-	-	**0.064**	-	-	-	-	-
d198	-	-	-	-	-	-	-	-
pcb442	-	-	-	-	-	-	-	-
rat783	-	-	-	-	-	-	-	-
d1291	-	-	-	-	-	-	-	-
pr2392	.	.	-	-	-	-	-	-
fnl4461	.	.	.	.	-	-	-	-
rl5915	.	.	.	.	.	-	-	-

TABLE 42
Measures of the benefits of applying thread control. The traveling salesman problem using a pair-wise-exchange neighborhood is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores			Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		
	CPU	GPU_{Tex}	GPU_{TexTC}	CPU	GPU_{Tex}	GPU_{TexTC}
eil101	+	+	+	+	+	+
d198	+	+	+	+	+	+
pcb442	+	+	+	+	+	+
rat783	+	+	+	+	+	+
d1291	+	+	+	+	+	+
pr2392	+	.	+	+	+	+
fnl4461	+	.	+	+	.	+
rl5915	+	.	+	+	.	+

Instance	Intel Xeon E5450 GeForce GTX 280 240 GPU cores			Xeon E5620 Tesla M2050 448 GPU cores		
	CPU	GPU_{Tex}	GPU_{TexTC}	CPU	GPU_{Tex}	GPU_{TexTC}
eil101	+	+	+	+	+	+
d198	+	+	+	+	+	+
pcb442	+	+	+	+	+	+
rat783	+	+	+	+	+	+
d1291	+	+	+	+	+	+
pr2392	+	+	+	+	+	+
fnl4461	+	+	+	+	+	+
rl5915	+	.	+	+	+	+

TABLE 43
Measures of the benefits of applying thread control. The traveling salesman problem using a pair-wise-exchange neighborhood is considered. Test of the null hypothesis of variances equality with the Levene's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU	GPU_{Tex}	CPU	GPU_{Tex}	CPU	GPU_{Tex}	CPU	GPU_{Tex}
	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}
eil101	+	+	+	+	+	+	+	+
d198	+	+	+	+	+	+	+	+
pcb442	+	+	+	+	+	+	+	+
rat783	+	+	+	+	+	+	+	+
d1291	+	+	+	+	+	+	+	+
pr2392	+	.	+	+	+	+	+	+
fnl4461	+	.	+	.	+	+	+	+
rl5915	+	.	+	.	+	.	+	+

TABLE 44
Measures of the benefits of applying thread control. The traveling salesman problem using a pair-wise-exchange neighborhood is considered. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU	GPU_{Tex}	CPU	GPU_{Tex}	CPU	GPU_{Tex}	CPU	GPU_{Tex}
	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}
eil101	-	**0.58**	-	-	-	**0.65**	-	-
d198	-	**0.62**	-	-	-	**0.60**	-	-
pcb442	-	-	-	-	-	-	-	-
rat783	-	**0.55**	-	-	-	-	-	-
d1291	-	**0.64**	-	-	-	-	-	-
pr2392	-	.	-	-	-	-	-	-
fnl4461	-	.	-	.	-	-	-	-
rl5915	-	.	-	.	-	.	-	-

TABLE 45
Measures of the benefits of applying thread control. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores			Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		
	CPU	GPU_{Tex}	GPU_{TexTC}	CPU	GPU_{Tex}	GPU_{TexTC}
73-73	+	+	+	+	+	+
81-81	+	+	+	+	+	+
101-117	+	+	+	+	+	+
201-217	+	+	+	+	+	+
401-417	+	+	+	+	+	+
601-617	+	+	+	+	+	+
801-817	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+

Instance	Intel Xeon E5450 GeForce GTX 280 240 GPU cores 32 GPU cores			Xeon E5620 Tesla M2050 448 GPU cores 128 GPU cores		
	CPU	GPU_{Tex}	GPU_{TexTC}	CPU	GPU_{Tex}	GPU_{TexTC}
73-73	+	+	+	+	+	+
81-81	+	+	+	+	+	+
101-117	+	+	+	+	+	+
201-217	+	+	+	+	+	+
401-417	+	+	+	+	+	+
601-617	+	+	+	+	+	+
801-817	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+

TABLE 46
Measures of the benefits of applying thread control. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of variances equality with the Levene's test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU	GPU_{Tex}	CPU	GPU_{Tex}	CPU	GPU_{Tex}	CPU	GPU_{Tex}
	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}
73-73	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+

TABLE 47
Measures of the benefits of applying thread control. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	Core 2 Duo T5800 GeForce 8600M GT 32 GPU cores		Core 2 Quad Q6600 GeForce 8800 GTX 128 GPU cores		Intel Xeon E5450 GeForce GTX 280 240 GPU cores		Xeon E5620 Tesla M2050 448 GPU cores	
	CPU	GPU_{Tex}	CPU	GPU_{Tex}	CPU	GPU_{Tex}	CPU	GPU_{Tex}
	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}	GPU_{TexTC}
73-73	-	-	-	-	-	-	-	-
81-81	-	-	-	-	-	-	-	-
101-117	-	-	-	-	-	-	-	-
201-217	-	-	-	-	-	-	-	-
401-417	-	-	-	-	-	-	-	-
601-617	-	-	-	-	-	-	-	-
801-817	-	-	-	-	-	-	-	-
1001-1017	-	-	-	-	-	-	-	-
1301-1317	-	-	-	-	-	-	-	-

TABLE 48
Measures of the benefits of generating the neighborhood on GPU on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	CPU	**Evaluation on GPU**				**Generation and evaluation on GPU**			
		GPU_{Tex}	process	transfers	kernel	GPU_{Tex}	process	transfers	kernel
73-73	+	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+	+
801-817	+	.	.	.	.	+	+	+	+
1001-1017	+	.	.	.	.	+	+	+	+
1301-1317	+	.	.	.	.	+	+	+	+

TABLE 49
Measures of the benefits of generating the neighborhood on GPU on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of variances equality with the Levene's test.

Instance	CPU - E/GPU_{Tex}	$E/GPU_{Tex} - EG/GPU_{Tex}$	
		transfers	kernel
73-73	+	+	+
81-81	+	+	+
101-117	+	+	+
201-217	+	+	+
401-417	+	+	+
601-617	+	+	+
801-817	.	.	.
1001-1017	.	.	.
1301-1317	.	.	.

TABLE 50
Measures of the benefits of generating the neighborhood on GPU on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	CPU - E/GPU_{Tex}	$E/GPU_{Tex} - EG/GPU_{Tex}$	
		transfers	kernel
73-73	-	-	-
81-81	-	-	-
101-117	-	-	-
201-217	-	-	-
401-417	-	-	-
601-617	-	-	-
801-817	.	.	.
1001-1017	.	.	.
1301-1317	.	.	.

TABLE 51
Measures of the benefits of generating the neighborhood on GPU on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of one is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	CPU	**Evaluation on GPU**				**Generation and evaluation on GPU**			
		GPU_{Tex}	process	transfers	kernel	GPU_{Tex}	process	transfers	kernel
73-73	+	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+	+

TABLE 52
Measures of the benefits of generating the neighborhood on GPU on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of one is considered. Test of the null hypothesis of variances equality with the Levene's test.

Instance	CPU - E/GPU_{Tex}	$E/GPU_{Tex} - EG/GPU_{Tex}$	
		transfers	kernel
73-73	+	+	+
81-81	+	+	+
101-117	+	+	+
201-217	+	+	+
401-417	+	+	+
601-617	+	+	+
801-817	+	+	+
1001-1017	+	+	+
1301-1317	+	+	+

TABLE 53
Measures of the benefits of generating the neighborhood on GPU on the GTX 280. The permuted perceptron problem using a neighborhood based on a Hamming distance of one is considered. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	CPU - E/GPU_{Tex}	$E/GPU_{Tex} - EG/GPU_{Tex}$	
		transfers	kernel
73-73	-	-	-
81-81	-	-	-
101-117	-	-	-
201-217	-	-	-
401-417	-	-	-
601-617	-	-	-
801-817	-	-	-
1001-1017	-	-	-
1301-1317	-	-	-

TABLE 54
Measures of the benefits of using the reduction operation on the GTX 280. The permuted perceptron problem is considered for two different neighborhoods using 100 hill climbing algorithms. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	n neighbors			$\frac{n\times(n-1)}{2}$ neighbors		
	CPU	GPU_{Tex}	GPU_{TexR}	CPU	GPU_{Tex}	GPU_{TexR}
73-73	+	+	+	+	+	+
81-81	+	+	+	+	+	+
101-117	+	+	+	+	+	+
201-217	+	+	+	+	+	+
401-417	+	+	+	+	+	+
601-617	+	+	+	+	+	+
801-817	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+

TABLE 55
Measures of the benefits of using the reduction operation on the GTX 280. The permuted perceptron problem is considered for two different neighborhoods using 100 hill climbing algorithms. Test of the null hypothesis of variances equality with the Levene's test.

Instance	n neighbors		$\frac{n\times(n-1)}{2}$ neighbors	
	CPU - GPU_{TexR}	GPU_{Tex} - GPU_{TexR}	CPU - GPU_{TexR}	GPU_{Tex} - GPU_{TexR}
73-73	+	+	+	+
81-81	+	+	+	+
101-117	+	+	+	+
201-217	+	+	+	+
401-417	+	+	+	+
601-617	+	+	+	+
801-817	+	+	+	+
1001-1017	+	+	+	+
1301-1317	+	+	+	+

TABLE 56
Measures of the benefits of using the reduction operation on the GTX 280. The permuted perceptron problem is considered for two different neighborhoods using 100 hill climbing algorithms. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	n neighbors		$\frac{n\times(n-1)}{2}$ neighbors	
	CPU - GPU_{TexR}	GPU_{Tex} - GPU_{TexR}	CPU - GPU_{TexR}	GPU_{Tex} - GPU_{TexR}
73-73	**0.064**	**0.059**	-	-
81-81	**0.058**	**0.057**	-	-
101-117	**0.071**	**0.060**	-	-
201-217	**0.062**	**0.063**	-	-
401-417	-	-	-	-
601-617	-	-	-	-
801-817	-	-	-	-
1001-1017	-	-	-	-
1301-1317	-	-	-	-

TABLE 57
Analysis of the time dedicated for each operation for 100 hill climbing algorithms. The permuted perceptron problem using a Hamming distance of two and the reduction operation are considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	GPU_{Tex}			GPU_{TexR}		
	process	transfers	kernel	process	transfers	kernel
73-73	+	+	+	+	+	+
81-81	+	+	+	+	+	+
101-117	+	+	+	+	+	+
201-217	+	+	+	+	+	+
401-417	+	+	+	+	+	+
601-617	+	+	+	+	+	+
801-817	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+

TABLE 58
Analysis of the time dedicated for each operation for 100 hill climbing algorithms. The permuted perceptron problem using a Hamming distance of two and the reduction operation are considered. Test of the null hypothesis of variances equality with the Levene's test.

Instance	GPU_{Tex} - GPU_{TexR}		
	process	transfers	kernel
73-73	+	+	+
81-81	+	+	+
101-117	+	+	+
201-217	+	+	+
401-417	+	+	+
601-617	+	+	+
801-817	+	+	+
1001-1017	+	+	+
1301-1317	+	+	+

TABLE 59
Analysis of the time dedicated for each operation for 100 hill climbing algorithms. The permuted perceptron problem using a Hamming distance of two and the reduction operation are considered. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	GPU_{Tex} - GPU_{TexR}		
	process	transfers	kernel
73-73	-	-	-
81-81	-	-	-
101-117	-	-	-
201-217	-	-	-
401-417	-	-	-
601-617	-	-	-
801-817	-	-	-
1001-1017	-	-	-
1301-1317	-	-	-

TABLE 60
Measures in terms of efficiency for a cluster of workstations. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	**Intel Xeon E5440 8 CPU cores**			**4 Intel Xeon E5440 32 CPU cores**			**11 Intel Xeon E5440 88 CPU cores**			**13 Intel Xeon E5440 104 CPU cores**		
	CPU	GPU_{Tex}	COW	CPU	GPU_{Tex}	COW	CPU	GPU_{Tex}	COW	CPU	GPU_{Tex}	COW
73-73	+	+	+	+	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+	+	+	+	+

TABLE 61
Measures in terms of efficiency for a cluster of workstations. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of variances equality with the Levene's test.

Instance	Intel Xeon E5440 8 CPU cores		4 Intel Xeon 2.83 Ghz 32 CPU cores		11 Intel Xeon E5440 88 CPU cores		13 Intel Xeon E5440 104 CPU cores	
	CPU COW	GPU_{Tex} COW	CPU COW	GPU_{Tex} COW	CPU COW	GPU_{Tex} COW	CPU COW	GPU_{Tex} COW
73-73	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+

TABLE 62
Measures in terms of efficiency for a cluster of workstations. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	Intel Xeon E5440 8 CPU cores		4 Intel Xeon 2.83 Ghz 32 CPU cores		11 Intel Xeon E5440 88 CPU cores		13 Intel Xeon E5440 104 CPU cores	
	CPU COW	GPU_{Tex} COW	CPU COW	GPU_{Tex} COW	CPU COW	GPU_{Tex} COW	CPU COW	GPU_{Tex} COW
73-73	-	-	-	-	-	-	-	-
81-81	-	-	-	-	-	-	-	-
101-117	-	-	-	-	-	-	-	-
201-217	-	-	-	-	-	-	-	-
401-417	-	-	-	-	-	-	-	-
601-617	-	-	-	-	-	-	-	-
801-817	-	-	-	-	-	-	-	-
1001-1017	-	-	-	-	-	-	-	-
1301-1317	-	-	-	-	-	-	-	-

TABLE 63
Analysis of the time dedicated for each operation for a cluster of workstations. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	Intel Xeon E5440 8 CPU cores			4 Intel Xeon 2.83 Ghz 32 CPU cores			11 Intel Xeon E5440 88 CPU cores		
	process	transfers	workers	process	transfers	workers	process	transfers	workers
73-73	+	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+	+

TABLE 64
Measures in terms of efficiency for workstations distributed in a grid organization. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	2 Intel Xeon QC E5440 8 CPU cores			5 machines 40 CPU cores			12 machines 96 CPU cores			14 machines 112 CPU cores		
	CPU	GPU_{Tex}	Grid	CPU	GPU_{Tex}	Grid	CPU	GPU_{Tex}	Grid	CPU	GPU_{Tex}	Grid
73-73	+	+	+	+	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+	+	+	+	+

TABLE 65
Measures in terms of efficiency for workstations distributed in a grid organization. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of variances equality with the Levene's test.

Instance	2 Intel Xeon QC E5440 8 CPU cores		5 machines 40 CPU cores		12 machines 96 CPU cores		14 machines 112 CPU cores	
	CPU Grid	GPU_{Tex} Grid	CPU Grid	GPU_{Tex} Grid	CPU Grid	GPU_{Tex} Grid	CPU Grid	GPU_{Tex} Grid
73-73	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+

TABLE 66
Measures in terms of efficiency for workstations distributed in a grid organization. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of the averages equality with the Student's t-test.

Instance	2 Intel Xeon QC E5440 8 CPU cores		5 machines 40 CPU cores		12 machines 96 CPU cores		14 machines 112 CPU cores	
	CPU Grid	GPU_{Tex} Grid	CPU Grid	GPU_{Tex} Grid	CPU Grid	GPU_{Tex} Grid	CPU Grid	GPU_{Tex} Grid
73-73	-	-	-	-	-	-	-	-
81-81	-	-	-	-	-	-	-	-
101-117	-	-	-	-	-	-	-	-
201-217	-	-	-	-	-	-	-	-
401-417	-	-	-	-	-	-	-	-
601-617	-	-	-	-	-	-	-	-
801-817	-	-	-	-	-	-	-	-
1001-1017	-	-	-	-	-	-	-	-
1301-1317	-	-	-	-	-	-	-	-

TABLE 67
Analysis of the time dedicated for each operation for workstations distributed in a grid organization. The permuted perceptron problem using a neighborhood based on a Hamming distance of two is considered. Test of the null hypothesis of normality with the Kolmogorov-Smirnov's test.

Instance	**2 Intel Xeon QC E5440 8 CPU cores**			**5 machines 40 CPU cores**			**12 machines 96 CPU cores**		
	process	transfers	workers	process	transfers	workers	process	transfers	workers
73-73	+	+	+	+	+	+	+	+	+
81-81	+	+	+	+	+	+	+	+	+
101-117	+	+	+	+	+	+	+	+	+
201-217	+	+	+	+	+	+	+	+	+
401-417	+	+	+	+	+	+	+	+	+
601-617	+	+	+	+	+	+	+	+	+
801-817	+	+	+	+	+	+	+	+	+
1001-1017	+	+	+	+	+	+	+	+	+
1301-1317	+	+	+	+	+	+	+	+	+

REFERENCES

[1] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[2] S. Ryoo, C. I. Rodrigues, S. S. Stone, J. A. Stratton, S.-Z. Ueng, S. S. Baghsorkhi, and W. mei W. Hwu, "Program optimization carving for gpu computing," *J. Parallel Distribributed Computing*, vol. 68, no. 10, pp. 1389–1401, 2008.

[3] J. D. Owens, M.Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[4] R. K. Ahuja, J. Goodstein, A. Mukherjee, J. B. Orlin, and D. Sharma, "A very large-scale neighborhood search algorithm for the combined through-fleet-assignment model," *INFORMS Journal on Computing*, vol. 19, no. 3, pp. 416–428, 2007.

[5] NVIDIA, *CUDA Programming Guide Version 4.0*, 2011.