J Supercomput (2016) 72:4204–4220
DOI 10.1007/s11227-016-1733-8

Application of distributed parallel computing for dynamic visual cryptography

Raimondas Čiegis[1] · Vadimas Starikovičius[2] · Natalija Tumanova[1] · Minvydas Ragulskis[3]

Published online: 4 May 2016

Abstract We consider one applied global optimization problem where a set of feasible solutions is discrete and very large. The goal is to find optimal perfect gratings, which can guarantee high quality and security of the visual cryptography method. A priori estimation techniques, such as branch and bound type methods, cannot be applied to exclude an essential part of elements from the feasible set. Thus, a full search is required to solve this global optimization problem exactly, which is very computationally demanding. A library of C++ templates is developed that allows its user to implement parallel master–slave algorithms for his/her application without any knowledge of parallel programming API (application programming interface). Design of the templates allows users to build a parallel solver using MPI (message passing interface) API or distributed computing application using BOINC (Berkeley open infrastructure for network computing) API from the same C/C++ code with implementation of application-specific tasks. We build parallel and distributed computing solvers for the considered optimization problem and present results of computational experiments using a computer cluster and BOINC project for volunteer computing. Heuristic methods are also considered as an alternative to the full search algorithm. Due to complicated conditions defining feasible solutions (perfect gratings), genetic algorithms cannot be used to solve this problem efficiently. We propose two memetic heuristic algorithms, when a basic stochastic or simplified full search algorithm is

✉ Vadimas Starikovičius
vs@vgtu.lt; vadimas.starikovcius@vgtu.lt

[1] Department of Mathematical Modeling, Vilnius Gediminas Technical University, Saulėtekio av. 11, 10223 Vilnius, Lithuania

[2] Laboratory of Parallel Computing, Vilnius Gediminas Technical University, Saulėtekio av. 11, 10223 Vilnius, Lithuania

[3] Research Group for Mathematical and Numerical Analysis of Dynamical Systems, Kaunas University of Technology, Studentu 50-222, 51368 Kaunas, Lithuania

combined with a local search algorithm. Parallel heuristic algorithms are also proposed and implemented. The efficiency and accuracy of heuristics are investigated and results of experiments are presented.

Keywords Parallel algorithm · Programming templates · Distributed computing · BOINC · Dynamic visual cryptography · Heuristics

1 Introduction

The recent activities to construct exascale and ultrascale distributed computational systems are opening a possibility to apply parallel and distributed computing techniques for applied problems which previously were considered as not solvable with standard computational resources. In this paper, we consider a global optimization problem, where a set of feasible solutions is discrete and very large. There is no possibility to apply well-known a priori estimation techniques to exclude an essential part of these elements from the computational analysis, e.g. applying branch and bound type methods [15]. Thus, a full search is required to solve this global optimization problem exactly. We note that in this paper some specific techniques (periodicity condition, mirror transformation) are applied to reduce the size of the set of feasible solutions, but they cannot change the general non-polynomial order of complexity of the algorithm.

The problem under consideration deals with the recently developed method for chaotic dynamic visual cryptography [24]. Visual cryptography is a cryptographic technique which allows visual information to be encrypted in such a way that the decryption can be performed by the human visual system, without any cryptographic computation. This concept was introduced by Naor and Shamir [22] in 1994 and it gained a significant popularity due to a number of important applications [14,26].

In this paper, we solve a global optimization problem, which solution guarantees a high quality and security of the considered method for dynamic visual cryptography. The size of the optimization problem, which can be solved exactly, essentially depends on the amount of available computational resources. Nowadays, the usual approach to solve such problems is to employ parallel and distributed computing techniques.

The presented global optimization problem can be solved in parallel using well-known master–slave programing model [17]. It is the main model used for solving various optimization problems in parallel [8,28]. It is important for our goals that master–slave model is well suited for heterogeneous and distributed computational resources [18,21].

To build parallel and distributed solvers for the given optimization problem, we have developed our own parallel programming templates (algorithmic skeleton) for the master–slave paradigm. The idea of high level parallel programing frameworks is still quite popular. It allows a user of such parallel programming templates or skeletons to obtain a parallel solver for his problem without any parallel programing. He needs only to implement application-specific parts of the algorithm, for example, for divide and conquer or branch and bound algorithm [9], Tabu search method [5]. A recent survey of parallel skeleton frameworks can be found in [13]. Our group has its own successful experience of development and usage of such parallel programing templates [3,27].

Design of programing templates developed in this work allows users to build a parallel application using MPI (message passing interface) API (application programming interface) [20] or distributed computing application using BOINC (Berkeley open infrastructure for network computing) API [2] from the same C/C++ code with implementation of application-specific tasks. The BOINC middleware was initially developed for well-known SETI@Home project [16]. Currently, it is the most popular open-source middleware for volunteer distributed computing. It is used for distributed computing applications in such diverse areas as astrophysics [1], molecular biology [4], medicine, climatology, linguistics, mathematics, and others. Millions of volunteers are providing resources of their computers to such scientific projects.

It is well known that heuristic algorithms present an alternative approach to full search algorithm, when approximate solution, which is obtained in reasonable time, produces sufficiently good results. This approach is currently used for the considered visual cryptography method [23]. However, more research is needed to investigate the quality of different heuristic algorithms. First results of this research were presented in [6]. We show that popular genetic algorithms cannot be used for the given problem. Due to complicated conditions defining feasible solutions (perfect gratings), the crossover of two perfect gratings is not producing a new feasible solution. A mutation procedure also in most cases produces non-perfect gratings.

The second objective of this paper is to investigate parallel heuristic algorithms for our global optimization problem. Two heuristics, as an alternative for the full search algorithm, are proposed. They are based on templates of memetic search algorithms for global optimization [11]. Our aim is to investigate the efficiency of such algorithms in the case when the set of feasible solutions is described by complicated non-local requirements.

The rest of this paper is organized as follows. In Sect. 2, the discrete global optimization problem is formulated. A set of feasible solutions (perfect gratings) is described and the optimality criterion for finding the optimal perfect gratings is defined. The parallel master–slave type algorithm for full search is presented in Sect. 3. Two heuristic algorithms, as an alternative for full search algorithm, are proposed in Sect. 4. In Sect. 5, we present developed programing templates, which allow users to build parallel MPI applications and distributed BOINC applications. Results of computational experiments are presented in Sect. 6. Some final conclusions are made in Sect. 7.

2 Discrete global optimization problem

First, we present the most important details on the chaotic dynamic visual cryptography method. A full description of this method can be found in [24]. The image hiding method is based not on the static superposition of shares, but on the time-averaging moiré gratings. It is important to note that using this method only one slide is generated. The secret image can be seen by a human visual system only when the encoded image is harmonically oscillated in a predefined direction at strictly defined amplitude of oscillation.

Function $F(x)$ defines a greyscale grating function if the following requirements are satisfied:

(i) The grating is a periodic function $F(x+\lambda) = F(x)$, where λ is the pitch of grating, and $0 \le F(x) \le 1$;
(ii) m-pixels n-level greyscale function $F_{mn}(x)$ is defined as:

$$F_{mn}(x) = \frac{y_k}{n}, \quad \frac{(k-1)\lambda}{m} \le x \le \frac{k\lambda}{m},$$

where $k = 1, \dots, m$, $0 \le y_k \le n$.

Not any grating function can be used in applications of dynamical visual cryptography. We will consider a subset P of perfect greyscale step functions; they satisfy the following additional requirements:

(1) The grating spans through the whole greyscale interval:

$$\min_{1\le k\le m} y_k = 0, \quad \max_{1\le k\le m} y_k = n.$$

(2) The average greyscale level in a pitch of the grating equals $n/2$:

$$\gamma := \frac{1}{m}\sum_{k=1}^{m} y_k = \frac{n}{2}.$$

(3) The "norm" of the greyscale grating function must be at least equal to the half of the norm of the harmonic grating:

$$\|F\| \ge \frac{1}{2}\|\widetilde{F}\| = \frac{1}{2\pi}, \quad \|F\| := \frac{1}{\lambda}\int_0^{\lambda}\left|F(x) - \frac{1}{2}\right| dx.$$

(4) The pitch of the grating λ must be strongly identifiable. The main peak of the discrete Fourier amplitude must be at least two times larger compared to all remaining Fourier modes:

$$\sqrt{a_1^2 + b_1^2} \ge 2\sqrt{a_j^2 + b_j^2}, \quad j = 2, 3, \dots, m-1,$$

where the function F is expanded into the Fourier truncated series:

$$F(x) = \frac{a_0}{2} + \sum_{j=1}^{m-1}\left(a_j \cos\frac{2\pi j x}{\lambda} + b_j \sin\frac{2\pi j x}{\lambda}\right).$$

The optimality criterion for finding the optimal perfect grating function is defined as:

$$\delta(F_{mn}^0) = \max_{F_{mn}\in P}\min_{s\in S_1}\left(\sigma\left(H_s(F_{mn}, \widetilde{\xi}_s)\right)\right), \tag{1}$$

where the standard deviation of a greyscale step grating function oscillated harmonically is given by (s is the oscillation amplitude):

$$\sigma\left(H_s(F_{mn}, \widetilde{\xi}_s)\right) = \frac{\sqrt{2}}{2}\sqrt{\sum_{j=1}^{m-1}(a_j^2 + b_j^2)J_0^2\left(2\pi js/\lambda\right)},$$

where J_0 is the Bessel function of the first type.

The minimization tasks for finding optimal oscillation amplitudes $s \in S_1$ are solved by the golden section search method [10].

Next, we present results of one application of this cryptography technique. The secret dichotomous image is presented in Fig. 1a. The encoded cover image (using the near-optimal moiré grating) is illustrated in Fig. 1b. The pitch of the moiré grating used for the secret image is 22 pixels. The pitch of the grating used for the background comprises 21 pixels. Phase matching and chaotic scrambling algorithms are used for embedding the secret image into the cover image [24]. The size of the cover image is 153.7 mm $\times$ 132 mm. The pitch of the moiré gratings in the areas occupied by the secret image and the background is 3.946 and 3.77 mm, respectively. Oscillations of the cover image according to the triangular waveform leak the secret at the amplitude of 1.97 mm in the time-averaged image (see Fig. 1c). Note that the near-optimal moiré grating is transformed into a uniform moiré fringe in the areas occupied by the secret image in the time-averaged image. Contrast enhancement techniques can be used to highlight the secret in the time-averaged image (see Fig. 1d) [25].

3 Parallel master–slave algorithm for the full search

In previous section, we have formulated global optimization problem, where a set of feasible solutions is discrete and very large. There is no possibility to apply some a priori estimation techniques to exclude an essential part of these elements from the computational analysis, e.g. applying branch and bound type methods [15]. Thus, a full search is required to solve this global optimization problems exactly.

This global optimization problem can be solved in parallel using well-known master–slave paradigm [17]. The parallel algorithm is defined as follows. A full set D of possible gratings is generated by master process and decomposed into uniform tasks, which are sent to slave processes. Slave processes receive these tasks from master process and apply the following two-step algorithm for each grating from the assigned task:

1. First, it is tested if all four conditions of perfect gratings are satisfied.
2. Second, for a perfect grating the value of the standard deviation of grating function (1) is computed and the local optimal value is updated if a greater value is obtained.

After finishing their tasks, slave processes send obtained local optimal values with the corresponding best perfect gratings back to the master process. Master process receives and compares new results with the best known perfect grating.

Note that all tasks are independent from each other, thus synchronization of tasks is fully avoided. Dynamic generation and scheduling of tasks of proper size should

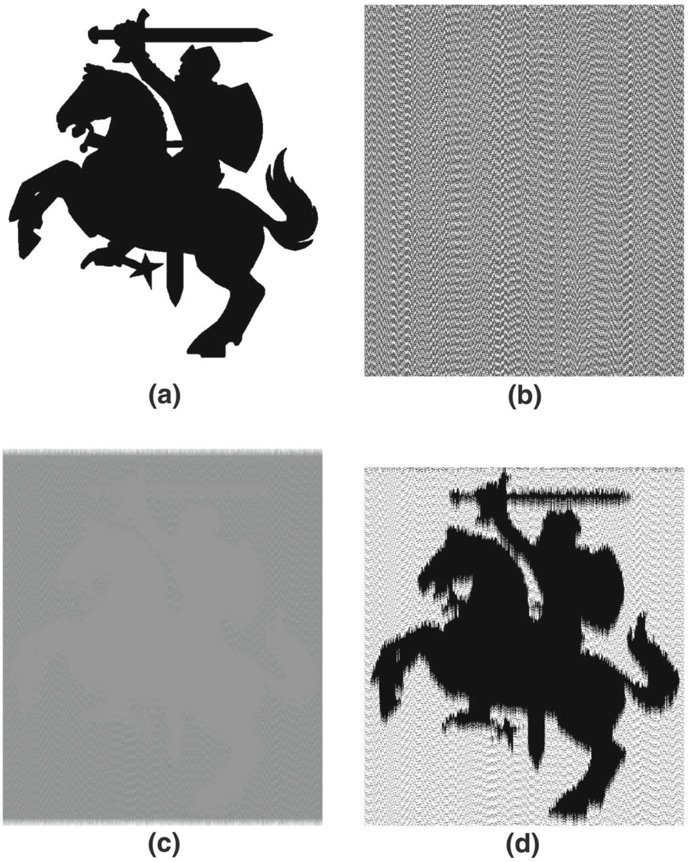

Fig. 1 Near-optimal moiré grating for visual decoding of the secret image. The secret dichotomous image (857 × 736 pixels) is illustrated in **a**; the encoded cover image is shown in **b**. The secret image is leaked in a form of time-averaged moiré fringes in **c**; contrast enhanced time-averaged image is shown in **d**

ensure a good performance of this parallel algorithm on any parallel or distributed computing system.

As it follows from the presented algorithm, the complexity of the full search sequential algorithm is of order

$$W = \mathcal{O}(n^m m \log m). \tag{2}$$

Here, the factor $m \log m$ arises due to the application of FFT algorithm. The obtained estimate shows a strong nonlinear dependence of W on the grid size m and the number of greyscale levels n.

For industrial visual cryptography applications, gratings with $12 \leq m \leq 25$ and $15 \leq n \leq 127$ are considered. To reduce the size of feasible solutions set D, two modifications of the basic algorithm are applied:

1. Due to the first condition of perfect gratings, a greyscale of the last pixel is fixed to $y_m = 0$.

2. The periodicity condition and mirror transformation of pixels are applied to exclude gratings, which were already tested during earlier stages of the full search algorithm.

It should be noted that such modifications still not change (reduce) asymptotics of the complexity of the proposed full search algorithm.

4 Heuristic algorithms

The presented parallel full search algorithm requires huge computational resources to determine the optimal perfect grating in reasonable time for any significant problem size. It is well known that for some similar discrete search problems, such as finding a key of cryptographical DES algorithm [7] or reconstruction of input data for a known hash function value [19]; the exact solution is the only meaningful solution, i.e. full search is required.

Luckily, in visual cryptography in many cases, it is sufficient to find a good approximation of the optimal perfect grating. Thus, heuristic methods can be considered as an alternative to the full search algorithm.

The problem of finding optimal perfect grating is formulated as a discrete global optimization problem. Thus, genetic evolutionary algorithms initially looked as a natural approach [12]. Some preliminary results on application of genetic algorithms are given in [6,23], where different modifications of the standard genetic algorithms are used to find quasi-optimal perfect gratings.

However, our analysis shows that genetic algorithms cannot be used to solve efficiently this problem. Due to complicated nonlocal conditions defining feasible solutions (perfect gratings), the crossover of two perfect gratings is not producing a new feasible solution (perfect grating). In most cases, a mutation procedure also is not giving a new perfect grating. Thus, both two most important features of any genetic algorithm are not working in proper way for finding optimal perfect gratings.

In this paper, we propose two memetic heuristic algorithms, when a basic stochastic or simplified full search algorithm is combined with a local search algorithm (see [11] for introduction into metaheuristic methods).

4.1 Full search for reduced size gratings combined with local search algorithm (FSRGLS)

Memetic algorithm FSRGLS is composed of two parts.

1. First, the full search algorithm is applied for reduced size gratings, e.g. gratings with $m/2$ pixels and $n/2$ greyscale levels can be investigated. A solution is computed using the proposed full search solver. Then, an initial approximation of a large-scale grating is computed by doubling the number of pixels and preserving the obtained relative values of greyscale levels.
   In cases when a size of the reduced problem is still too large, a parallel version of the full search solver is used.

2. Second, a new set of feasible solutions is defined: each pixel can take a greyscale level $\tilde{y}_j$ in a small neighbourhood of y_j:

   $$DN(k) = \{(\tilde{y}_1, \dots, \tilde{y}_m): \quad -k \le y_j - \tilde{y}_j \le k, \quad j = 1, \dots, m\}.$$

   It follows that the size of this set is $O((2k+1)^m)$. In all numerical experiments, we have restricted computations to the minimum value of stencil $k = 1$. Thus, the complexity of one iteration of local search algorithm is defined by

   $$W = 3^m m \log m. \qquad (3)$$

   The second part of the proposed heuristic algorithm is also fully parallel. Scheduling of tasks for parallel processes is implemented using the same template as for parallel full search algorithm.
   The new quasi-optimal grating is searched in $DN(k)$. If a better solution is obtained, then a set of feasible solutions is recomputed around a new optimal solution $y_j = \tilde{y}_j$, $j = 1, \dots, m$ and the local search step is repeated.

4.2 Random search method combined with local search algorithm (RSLS)

In this method, we employ a different first step.

1. First, we sample in random a set of gratings (stochastic search algorithm) using a generator of pseudo-random numbers. The requirements of perfect gratings are checked for any generated grating. The obtained set of perfect gratings makes a subset of all feasible gratings.
2. Second, for a subset of generated perfect gratings, the local search algorithm of FSRGLS is applied. This part of the algorithm is fully parallel.

In computational experiments, we have tested the quality of solutions obtained using both heuristics. Results are presented in Sect. 6.3.

5 Master–slave parallel programing templates for MPI and BOINC

To build parallel and distributed applications for our problem, we develop our own parallel programming templates (algorithmic skeleton) for the master–slave paradigm. The idea of such templates or parallel skeletons is to provide for user a possibility to create parallel applications without any parallel programming and knowledge of parallel programing API [3,9,13,27]. Such templates are reducing the time and efforts needed for development of new parallel applications even for the experienced parallel programmer.

The main idea of our programing templates is to obtain a distributed computing application and parallel application from the same C/C++ code, which needs to be provided by user and implements problem-specific parts of whole algorithm without parallel programing API. In accordance with the master–slave paradigm [17], user needs to implement reading of the problem input, consecutive generation of single jobs (tasks), solving of the single job (task), processing or merging of obtained results.

Let us now formulate the key features of developed master–slave parallel programing templates:

– Design of the templates allows to build a parallel application using MPI API [20] or distributed computing application using BOINC API [2] using the same C/C++ code with implementation of problem-specific tasks.
– The templates are built as a hierarchy of C++ classes. The basic classes implement the basic functionality of any master–slave algorithm and ensure its workflow including communication between the master and slave processes.
– The problem-specific parts of the algorithm need to be implemented in descendant classes and placed in appropriate virtual functions.
– Input parameters and results of the job are exchanged between the master and slave processes using input and output files.

The usage of technology based on input and output files is not as efficient as a direct message passing between processes. However, the performance overhead is negligible for the coarse grained jobs. This requirement is satisfied for our problem, because we can easily adjust the size of a single job. In turn, such an approach significantly simplifies the template and allows efficient communication of input and output data between the master and slave processes without problem-specific parallel programming.

Such an approach also allows the implementation of distributed computing applications. Currently, our programing tool allows easy and quick development of distributed application for volunteer computing project based on the Berkeley open infrastructure for network computing (BOINC) [2], which is the most popular middleware for volunteer distributed computing. Using our C++ templates, application for BOINC project can be developed without any knowledge of BOINC API. Moreover, MPI version of application solver is very useful in testing and debugging implementations of problem-specific tasks.

Problem-specific tasks are separated and implemented in the following C++ classes:

– *WorkGenerator* class
  It reads the problem-specific input in its constructor, generates and writes to the properly named file the input for the next job by consecutive calls to the problem-specific function
  GenerateInputForNewJob(FILE *jobInputFile),
  which must be provided by the application developer.
– *ClientApplication* class
  It reads the input file, which was generated by *WorkGenerator*, solves the job, and writes the results to output file by calling problem-specific function
  SolveSingleJob(const char* inputFileName, const char* outputFileName),
  which must be provided by the application developer.
– *ResultsAssimilator* class
  It is the processing results of the single job and merging them with previously obtained results by calling problem-specific function
  AssimilateResults(FILE *jobResultsFile),
  which must be provided by the application developer.

Results obtained from the BOINC project users can be corrupted due to the different reasons. They need to be validated before the assimilation—merging with previously obtained results. This is a responsibility of BOINC project developer. In our programing template, we have included *ResultsValidator* class, which problem-specific functionality should be implemented in function *ValidateResults*(FILE *jobInputFile, FILE *jobResultsFile).

Note that this implementation can be tested on the computer cluster with parallel MPI application before the deployment of distributed application to the BOINC project.

6 Computational experiments

This section consists of four parts. First, we investigate the efficiency and scalability of the parallel MPI implementation of full search algorithm on computer cluster. Then, we investigate the performance of distributed application on our BOINC project. Next, we study the quality of two proposed heuristic algorithms. Finally, we investigate the scalability of the parallel MPI implementation of heuristic FSRGLS algorithm.

6.1 Performance of parallel MPI implementation of the full search algorithm

Parallel numerical tests were performed on the computer cluster "HPC Sauletekis" (http://www.supercomputing.ff.vu.lt) at the High Performance Computing Center of Vilnius University, Faculty of Physics. We have used up to eight nodes with Intel® Xeon® processors E5-2670 with 16 cores (2.60 GHz) and 128 GB of RAM per node. Computational nodes are interconnected via the InfiniBand network.

The parallel MPI implementation of the full search algorithm is built from master–slave programing templates described in Sect. 5. According to the design of templates, we have implemented the problem-specific parts of the parallel algorithm described in Sect. 3.

To investigate the parallel performance of such implementation, we have restricted to the analysis of quite small benchmark problem. We have solved the optimization problem for $m = 10, n = 15$. In Table 1, we present the total wall time $T_{s,p\times c}$ in seconds, when parallel computations are performed on a cluster with p nodes and c cores per node, and s slaves have solved computational tasks. The master process is responsible for generation and distribution of jobs and accumulation of results from slave processes. On computer cluster with Portable Batch System (PBS) job management, a separate core is allocated and used to run this part of the parallel algorithm. Also, we present the values of parallel algorithmic speed-up $S_s = T_{s,p\times c}/s$.

It follows from the presented results that the scalability of the parallel algorithm and its implementation using developed templates are very good. Some degradation of the efficiency for the largest numbers of parallel processes is explained by the load imbalance for this relatively small problem. The total number of generated tasks is 801. Note that granularity (the size) of the task is easily adjustable in our algorithm.

In Table 1, we also show the efficiency of utilization of hyperthreading mode on the computing node with 16 physical and 32 logical cores. Slight performance gains

Table 1 The total wall time $T_{s,p\times c}$ and speed-up S_s values for solving the grading optimization problem with $m = 10$, $n = 15$ by parallel full search solver

	$1, 1\times 2$	$2, 1\times 3$	$4, 1\times 5$	$8, 1\times 9$	$15, 1\times 16$	$16, 1\times 17$	$31, 1\times 32$
$T_{s,p\times c}$	7512	3781	1876	937	499.6	488.5	464.8
S_s	1	1.987	4.00	8.01	15.03	15.4	16.2
	$31, 2\times 16$	$48, 3\times 16$	$63, 4\times 16$	$79, 5\times 16$	$95, 6\times 16$	$127, 8\times 16$	
$T_{s,p\times c}$	245.5	165.6	124.5	100.8	86.4	65.6	
S_s	30.6	45.4	60.3	74.5	86.9	114.4	

can be observed using 17 and 32 parallel processes on the node: compare $T_{15,1\times 16}$ to $T_{16,1\times 17}$ and $T_{31,1\times 32}$.

The obtained estimate (2) of the complexity of full search algorithm allows us to estimate quite accurately the total computation times required to find optimal perfect gratings of increased sizes. For example, using results of previous computational experiments we get prediction that a problem with $m = 14$, $n = 19$ on 8×16 cluster will be solved in $T = 1478$ days. We have solved this problem using our BOINC project (for more details see the next subsection) in 73 days. The optimal perfect grating

$$F^0_{14,19} = (18,\ 17,\ 15,\ 18,\ 18,\ 18,\ 0,\ 0,\ 0,\ 18,\ 3,\ 1,\ 0)$$

is used as a reference solution for testing the efficiency and accuracy of heuristic search algorithms FSRGLS and RSLS in Sect. 6.3. The quality of the obtained optimal perfect grating is equal to $\delta(F^0_{14,19}) = 0.067814$.

6.2 Performance of distributed BOINC implementation of the full search algorithm

As described in Sect. 5, we can obtain from our parallel programing templates not only a parallel solver built on MPI, but also a distributed computing application built on BOINC platform. This open source platform is commonly used for the creation of volunteer computing projects. Such projects attract volunteers from all over the world to donate their computational resources, usually idle computers, for the participation in scientific computations using developed distributed applications. This technology allows to accumulate potentially very significant computational resources and tackle otherwise unsolvable problems.

We have built the distributed computing application for finding optimal perfect gratings using developed programing templates. According to the design of our templates, work generator, client application, result assimilator and validator are reusing the problem-specific code, which was written and tested developing parallel MPI solver. The developed visual cryptography application was deployed to our volunteer computing project called VGTU project@Home at http://boinc.vgtu.lt.

Currently, our project has accumulated over 1340 volunteers with 5057 computers, which have registered on the project page and already earned some credits, i.e. performed some work solving tasks sent from our project server. The important property of volunteer computing projects is a highly variable speed of computations, which depends on the number of currently active computers. Our work generator dynamically adapts to the available computational resources—computers periodically asking for tasks to solve.

To determine current and potential capability of this computing approach, we have started from the problem with $m = 13$ pixels and $n = 17$ greyscale levels. On our cluster, this problem would be solved with 8 nodes in 17.7 days. At VGTU project@Home, it was solved in 7 days. 168 users have participated in these computations with 700 computers. During these experiments, the aggregate speed of computations reached 625 GigaFLOPs. Again, we have noticed that the speed of computations was not constant. All tasks were generated and sent to the users in 34 h. 80 % of the tasks were solved in first three days.

We note that recently the computations of our project have attracted much more participants and the aggregated speed reached 4.5 TeraFLOPS. As the number of active users and their computers increases, we are going to solve increasable bigger problems. However, for the real size problems, heuristic algorithms need to be used so far. The speed of computations can also be increased with the development and deployment of GPU application version for visual cryptography problem.

6.3 Heuristic search algorithms

As it was already noted, the developed full search algorithm requires huge computational resources and finding optimal real size gratings is still a great challenge even for ultrascale distributed computational systems. Thus, the alternative approach based on heuristic algorithms is very important for our problem. For considered problem, it is often sufficient to find a good approximation of the optimal solution.

In this subsection, we present results of computational experiments for two heuristic search algorithms proposed in Sect. 4. First, we have tested the accuracy and convergence of these heuristics. We have used two benchmark problems with known optimal perfect gratings. The first benchmark problem is obtained by finding the optimal perfect grating for $m = 10$ pixels and $n = 15$ greyscale levels. This problem is solved exactly using the parallel full search solver. The obtained reference solution is

$$F^0_{10,15} = (13,\ 0,\ 9,\ 14,\ 10,\ 14,\ 10,\ 0,\ 0,\ 0)$$

and the standard deviation is equal to $\delta(F^0_{10,15}) = 0.060384$. This problem is solved on 8×16 cluster in $T_{127,8\times 16} = 65.5$ s (see Table 1).

In the first step of FSRGLS heuristic solver, a reduced size optimal perfect grating is computed for $m = 5$ pixels and $n = 8$ greyscale levels:

$$F^0_{5,8} = (7,\ 7,\ 1,\ 2,\ 0).$$

Then, applying a local search algorithm an approximation of the optimal perfect grating is computed for which the standard deviation is equal to $\delta(F^1_{10,15}) = 0.05755$. It is important to note that CPU time for the full application of sequential FSRGLS solver is only 2 s.

Next, the same problem is solved using RSLS heuristic algorithm. In the first step, sets of 5000, 10,000 and 20,000 perfect gratings are generated using a generator of pseudo-random numbers. This step required 2, 4 and 8 s of CPU time, respectively. Then, applying a local search algorithm approximations of the optimal perfect grating is computed with the following standard deviations:

$$\delta(F^{5000}_{10,15}) = 0.05985, \quad \delta(F^{10,000}_{10,15}) = 0.05725, \quad \delta(F^{20,000}_{10,15}) = 0.06018.$$

Comparing results of the FSRGLS and RSLS heuristic solvers, we can make a conclusion that for this benchmark problem the RSLS solver finds a better approximation of the optimal grating, but the multiscale FSRGLS solver required less CPU time.

The second benchmark problem is obtained by solving a larger grating with $m = 14$ pixels and $n = 19$ greyscale levels. The optimal perfect grating was obtained using the parallel full search solver on BOINC project in 73 days. On 8×16 cluster, this problem would be solved in around 1478 days. We remind that the obtained reference solution is

$$F^0_{14,19} = (18,\ 17,\ 15,\ 18,\ 18,\ 18,\ 0,\ 0,\ 0,\ 18,\ 3,\ 1,\ 0)$$

and the standard deviation is equal to $\delta(F^0_{14,19}) = 0.067814$. In the first step of FSRGLS heuristic solver, a reduced size optimal perfect grating is computed for $m = 7$ pixels and $n = 10$ greyscale levels:

$$F^0_{7,10} = (9,\ 8,\ 9,\ 1,\ 1,\ 3,\ 0)$$

and the standard deviation $\delta(F^0_{14,19}) = 0.058469$. Then, a local search algorithm is applied and after 12 iterations the optimal perfect grating is computed. It is important to note that CPU time for the full application of sequential FSRGLS solver is only 26 s.

Next, the same problem is solved using RSLS heuristic algorithm. In the first step, a set of 40,000 perfect gratings is generated using a generator of pseudo-random numbers. This step required 129 s of CPU time and the best standard deviation is equal to $\delta(F^{40,000}_{14,19}) = 0.057186$.

Then, a local search algorithm is applied for two initial approximations of gratings: the best one and a grating taken in random from the set of 20 best perfect gratings generated in the first step. After 6 and 11 iterations, we have computed approximations of the optimal perfect grating with the following standard deviations:

$$\delta(F^1_{14,19}) = 0.062554, \quad \delta(F^2_{14,19}) = 0.066791.$$

The CPU time for the local search algorithm of sequential RSLS solver is 19 and 36 s. It is interesting to note that a better approximation is computed starting from a worser initial grating.

Comparing results of the FSRGLS and RSLS solvers, we can make a conclusion that both heuristics can give quite good approximations of the optimal perfect gratings after reasonable amount of CPU time. The FSRGLS solver has required incomparable less CPU time and this property is explained by the fact that a structure of the initial approximation obtained after solving a reduced size problem mimics the structure of the optimal large size grating.

An increased amount of CPU time for the RSLS local search solver in comparison with FSRGLS solver is explained by a more random distribution of maximum and minimum greyscale levels in the initial gratings generated by the pseudo-random algorithm. The quality of obtained gratings is not so good and the standard deviation value is increasing very slowly when the number of generated gratings is increased.

So far, it is difficult to assess, which heuristic is more efficient. More research on this topic is needed with not only mathematical, but also engineering interpretation of the quality of obtained gratings.

6.4 Performance of parallel MPI implementation of the heuristic FSRGLS algorithm

Next, we have used our parallel programing templates for a quick development of parallel MPI-based versions of the FSRGLS and RSLS solvers. In this section, we present scalability tests of the obtained parallel FSRGLS solver. We have solved the optimization problem with $m = 20$ pixels and $n = 29$ greyscale levels; for this problem, the optimal perfect grating is not known. Using the optimal perfect grating for $m = 10, n = 15$, the following initial grating is used for a local search algorithm:

$$F^0_{20,29} = (26,\ 26,\ 0,\ 0,\ 18,\ 18,\ 28,\ 28,\ 20,\ 20, 28,\ 28,\ 20,\ 20,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0).$$

The stencil width $k = 1$ is defined in all computational tests. Then, a local search algorithm is applied and after 19 iterations an approximation of the optimal perfect grating is computed

$$F_{20,29} = (28,\ 28,\ 0,\ 0,\ 10,\ 28,\ 28,\ 28,\ 28,\ 28,\ 15, 28,\ 23,\ 8,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0),$$

for which the standard deviation is $\delta(F_{20,29}) = 0.067549$ (compare it with $\delta(F_{10,15}) = 0.060384$).

In Table 2, we present the total wall time $T_{s,p\times c}$ in seconds, when parallel computations are performed on a cluster with p nodes and c cores per node, and s slaves have solved computational tasks. Also, we present the values of parallel algorithmic speed-up S_s.

It follows from the presented results that the scalability of the parallel algorithm and its implementation using developed templates are very good.

Table 2 The total wall time $T_{s,p\times c}$ and speed-up S_s values for solving the grading optimization problem with $m = 20$, $n = 29$ by parallel FSRGLS solver

	$1, 1 \times 2$	$2, 1 \times 3$	$4, 1 \times 5$	$8, 1 \times 9$	$16, 1 \times 17$
$T_{s,p\times c}$	241.2	120.6	60.60	30.55	16.22
S_s	1	1.999	3.981	7.897	14.87

7 Conclusions

In this paper, we have described the parallel programing templates (algorithmic skeleton) for the implementation of parallel master–slave type of algorithms. Only implementation of the problem-specific tasks must be provided by user. In implementation of these tasks, the user need not to use any parallel programing API. The user of our templates can generate parallel solvers using MPI API or distributed computing applications using BOINC API from the same problem-specific C/C++ code. This allows the development and testing of complicated distributed BOINC applications in more convenient MPI environment.

Using our templates, we develop first a parallel MPI implementation of full search algorithm for global optimization problem of dynamic visual cryptography. The results of computational experiments have confirmed good scalability of parallel full search algorithm and its implementation.

Next, we have automatically built a distributed computing application for full search algorithm on BOINC platform. Developed distributed application for dynamic visual cryptography has no scalability problems with the increasing number of computing hosts. The size of single task (workunit) can be easily adjusted.

Results of computational experiments have confirmed that BOINC platform presents an attractive alternative to supercomputer systems. Developed programing templates can significantly reduce the time and efforts needed for the development of BOINC applications, especially, for the problems with available sequential codes.

The size of the considered discrete global optimization problem is huge even for the modern high performance computing systems. Thus, heuristic methods are also considered as an alternative to the full search algorithm. We have found that standard genetic heuristic algorithms are not efficient for the considered problem. Poor performance of genetic algorithms is caused by the fact that mutation and crossover processes of two perfect gratings usually are not producing a new perfect grating. But exactly this step is the most important for obtaining efficient genetic algorithms solving discrete global optimization problems.

In this paper, two memetic heuristic algorithms are proposed. At the first stage, some approximations of gratings are computed by simple global optimization algorithms. At the second stage, local improvement algorithms are applied. For the local optimization, the modification of the full search algorithm is proposed. Performed tests show that both heuristics can give quite good approximations of the optimal perfect gratings after reasonable amount of CPU time. So far, it is difficult to assess, which heuristic is more efficient. More research on this topic is needed with not only mathematical, but also engineering interpretation of the quality of obtained gratings.

Parallel heuristic solvers again are obtained using the described parallel templates. Results of computational experiments confirm good scalability of obtained parallel solvers.

Acknowledgments The work presented in this paper has been partially supported by EU under the COST programme Action IC1305, 'Network for Sustainable Ultrascale Computing (NESUS)'. This research was also funded by a Grant NUMMOD (No. MIP-074/2015) from the Research Council of Lithuania. Computations were performed on resources at the High Performance Computing Center "HPC Sauletekis" in Vilnius University Faculty of Physics. The authors would like to thank the referees for their comments which are helpful and constructive.

References

1. Allen B et al (2013) The Einstein@Home search for radio pulsars and PSR J2007+2722 discovery. Astrophys J 773(2):91
2. Anderson DP (2004) Boinc: a system for public resource computing and storage. In: Proceedings of the 5th IEEE/ACM international workshop on grid computing, pp 1–7
3. Baravykaite M, Čiegis R (2007) An implementation of a parallel generalized branch and bound template. Math Model Anal 12(3):277–289
4. Beberg AL, Ensign DL, Jayachandran G, Khaliq S, Pande VS (2009) Folding@Home: lessons from eight years of volunteer distributed computing. Parallel and Distributed Processing Symposium, International, pp 1–8
5. Blesa MJ, Hernàndez L, Xhafa F (2002) Parallel processing and applied mathematics: 4th international conference, PPAM 2001, Poland, September 9–12, 2001 Revised Papers, chap Parallel skeletons for Tabu search method based on search strategies and neighborhood partition, pp 185–193
6. Čiegis R, Starikovičius V, Tumanova N, Ragulskis M, Palivonaite R (2015) Distributed parallel computing for visual cryptography algorithms. In: Proceedings of the second international workshop on sustainable ultrascale computing systems (NESUS 2015), pp 23–28
7. Davies DW, Price WL (1989) Security for computer networks, 2nd edn. Wiley, New York
8. Depolli M, Trobec R, Filipič B (2013) Asynchronous master-slave parallelization of differential evolution for multi-objective optimization. Evol Comput 21(2):261–291
9. Dorta I, Leon C, Rodriguez C, Rojas A (2003) Parallel skeletons for divide-and-conquer and branch-and-bound techniques. In: Proceedings of 11th Euromicro conference on parallel, distributed and network-based processing (Euro-PDP'03), IEEE, pp 292–298
10. Gerald CF, Wheatley PO (2004) Applied numerical analysis. Addison-Wesley, Boston
11. Glover F, Kochenberger GA (2003) Handbook of metaheuristics, international series in operations research and management science. Springer, Berlin
12. Goldberg D (2002) The design of innovation: lessons from and for competent genetic algorithms. Kluwer Academic Publishers, Dordrecht
13. Gonzalez-Velez H, Leyton M (2010) A survey of algorithmic skeleton frameworks: high-level structured parallel programming enablers. Softw Pract Exp 40(12):1135–1160
14. Hegde C, Manu S, Shenoy PD, Venugopal K, Patnaik L (2008) Secure authentication using image processing and visual cryptography for banking applications. In: ADCOM 2008. 16th international conference on advanced computing and communications, IEEE, pp 65–72
15. Horst R, Pardalos PM, Thoai NV (2000) Introduction to global optimization, 2nd edn. Kluwer Academic Publishers, Dordrecht
16. Korpela EJ (2012) SETI@Home, BOINC, and volunteer distributed computing. Annu Rev Earth Planet Sci 40(1):69–87
17. Kumar V, Grama A, Gupta A, Karypis G (1994) Introduction to parallel computing: design and analysis of algorithms. Benjamin/Cummings, San Francisco
18. Lewis A, Mostaghim S, Scriven I (2009) Asynchronous multi-objective optimisation in unreliable distributed environments. In: Lewis A, Mostaghim S, Randall M (eds) Biologically-inspired optimisation methods: parallel algorithms, systems and applications, studies in computational intelligence, vol 210. Springer, Heidelberg, pp 51–78

19. Menezes AJ, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, Boca Raton
20. Message Passing Interface Forum (1995) MPI: a message passing interface standard. http://www.mpi-forum.org
21. Mostaghim S, Branke J, Lewis A, Schmeck H (2008) Parallel multi-objective optimization using master–slave model on heterogeneous resources. In: Evolutionary computation, 2008. CEC 2008. IEEE world congress on computational intelligence, pp 1981–1987
22. Naor M, Shamir A (1995) Visual cryptography. In: De Santis A (ed) Advances in cryptology—EUROCRYPT'94, lecture notes in computer science, vol 950. Springer, Berlin, pp 1–12
23. Palivonaite R, Fedaravicius A, Aleksa A, Ragulskis M (2013) Near-optimal moire grating for chaotic dynamic visual cryptography. In: Zaman H, Robinson P, Olivier P, Shih T, Velastin S (eds) Advances in visual informatics, lecture notes in computer science, vol 8237. Springer International Publishing, Switzerland, pp 48–58
24. Ragulskis M, Aleksa A (2009) Image hiding based on time-averaging moiré. Opt Commun 282(14):2752–2759
25. Ragulskis M, Aleksa A, Maskeliunas R (2009) Contrast enhancement of time averaged fringes based on moving average mapping functions. Opt Lasers Eng 47(7–8):768–773
26. Ross A, Othman A (2011) Visual cryptography for biometric privacy. IEEE Trans Inform Forensics Secur 6(1):70–81
27. Starikovičius V, Čiegis R, Iliev O (2011) A parallel solver for the design of oil filters. Math Model Anal 16(2):326–341
28. Talbi EG, Mostaghim S, Okabe T, Ishibuchi H, Rudolph G, Coello Coello CA (2008) Parallel approaches for multiobjective optimization. In: Branke J, Deb K, Miettinen K, Slowinski R (eds) Multiobjective optimization: interactive and evolutionary approaches, lecture notes in computer science, vol 5252. Springer, Heidelberg, pp 349–372