Application of Grid Computing for Designing a Class of Optimal Periodic Nonuniform Sampling Sequences

Andrzej Tarczynski¹, Tamas Kiss², Gabor Terstyanszky², Thierry Delaitre², Dongdong Qu¹, Stephen Winter²

¹Department of Electronic Systems and Software Engineering, University of Westminster
115 New Cavendish Street, London, W1W 6UW
e-mail: tarczya@westminster.ac.uk

²Centre for Parallel Computing, University of Westminster
115 New Cavendish Street, London, W1W 6UW
e-mail: kisst@wmin.ac.uk

Abstract—Designing periodic nonuniform sampling sequences for digital alias free signal processing is a computationally extensive problem where sequential single computer based solutions could easily run for days or even weeks. In order to reduce computation time, the sequential algorithm has been parallelised making it possible to execute parts of the calculations on different nodes of a computational Grid at the same time, reducing the overall runtime of the application. This paper presents and compares two different Grid based implementations representing the two main Grid approaches at the moment. The first solution utilizes a production Grid environment based on GEMLCA and the P-Grade Grid portal, and the second represents a BOINC desktop Grid based solution.

I. INTRODUCTION

In traditional Digital Signal Processing (DSP) analysis, the processed signals are sampled at uniformly distributed time instants. While use of such sampling schemes has obvious advantages like cyclostationarity of signal processing systems or facilitating use of efficient processing algorithms e.g. Fast Fourier Transform (FFT), uniform sampling suffers from a well-known limitation that prevents it from use in frequency ranges wider than half of the sampling rate. This limitation, known as aliasing, does not allow telling apart signal’s sinusoidal components if the sum or difference of their frequencies is an integer multiple of the sampling frequency. This fact is illustrated in Figure 1. We show there a sequence of noisy samples of a sinusoidal signal. The samples are taken at 200MSps rate (sample time is 5 ns). In turn, we attempt to fit sinusoids of frequencies 40 MHz and 160MHz into the acquired samples. In both cases we use the least squares method to get the best fit. Both sinusoids fit the data equally well. Therefore, it is impossible to tell which of them, if any, has been sampled.

The classical way of preventing aliasing is to establish which frequency ranges could be present in the processed signal. We refer to these ranges as the signal spectral support. The support should be conservative, i.e. no frequency present in the signal should be neglected. Once the spectral support is established the sampling rate is chosen in such a way that no two frequencies present in the support could become aliases of each other. If the spectral support is chosen very conservatively, then the resultant sampling rate is high in comparison with the Landau rate [1] – the theoretically lowest sampling rate that allows perfect signal reconstruction. Landau rate equals the double-sided bandwidth of the signal. In some cases, in order to preserve signal reconstructability, sampling at the Landau rate may require taking the samples at nonuniformly distributed time instants. Therefore, a more common approach is to sample signals at the smallest possible uniform sampling rate. Such sampling frequency never exceeds the Nyquist rate – twice the highest frequency present in the signal.

Use of excessive sampling rates (super-Landau or even super-Nyquist) is often acceptable, particularly when dealing with low-frequency signals. However, in some cases, e.g. processing of gigahertz signals, it could lead to solutions that are either economically or technically not viable. Digital alias-free signal processing (DASP) is an approach that offers effective solutions to processing signals with conservatively estimated spectral support. It uses carefully designed low-rate nonuniform sampling schemes and appropriate to them processing algorithms. To illustrate how nonuniform sampling could be used for suppressing aliasing we repeat the experiment shown in Figure 1. This time however, we use nonuniform sampling to collect data. The distances between consecutive sampling instants have been selected randomly between 5ns and 10 ns. The results are shown in Figure 2. Now it is obvious that the samples have been taken of the 160MHz sinusoid as it fits the data much better than the 40MHz sinusoid. Interestingly, this result has been reached even though the sampling rate is lower than the one used in the uniform case.
DASP-type solutions reported in research literature rely on either random sampling [2]-[4] or on periodic nonuniform sampling (PNS) [5]-[7]. In this paper we revisit the problem of designing PNS sequences for DASP applications that has been originally presented in [7]. This time however, we concentrate on computational aspects of the problem. We show that selection of the optimal sampling sequence is a computationally expensive problem and solutions based on a single computer may require long waiting times before the results are produced.

In order to make search for optimal solution more practical, we have taken a number of steps to reduce the total workload and implemented the optimization procedure in a grid environment to speed up the calculations.

This paper is organized as follows. In the next section we give brief description of the design of the PNS method that was originally introduced in [7]. Then we look at the numerical aspects of designing the sequence and show some simulation results. In section IV we describe how the optimization procedure has been implemented in a grid environment. The results of testing the new implementation and analysis of how much timing is improved are presented in section V. We conclude the paper in section VI and discuss briefly further improvements that can be introduced to grid implementations.

II. PNS FOR DASP

A. General Analysis of PNS

PNS is characterized by the layout of sampling instants inside the first sequence’s period $T_1$: \{τ_1, ..., τ_N\}. The remaining sampling instants are: $\tau_m = \tau_{N + \tau} = \tau_n + kT$ where $\tau_n$ is a sampling instant from the first period and $k$ is an integer. When $N = 1$ PNS becomes a classical uniform sampling sequence. In order to find the relationship between the spectra of continuous-time and discrete-time (sampled) signals we introduce sampling function $s(t)$ that represents the sequence $\tau_m$ as a train of Dirac’s impulses placed at $\tau_m$.

$$s(t) = \sum_{m=-\infty}^{\infty} \delta(t - \tau_m)$$  \hspace{1cm} (1)

Let $x(t)$ be a continuous-time signal and $x_d(t)$ a discrete-time signal obtained from sampling $x(t)$ at $\tau_m$:

$$x_d(t) = s(t)x(t)$$  \hspace{1cm} (2)

If the sampling sequence suppressed aliasing then the spectrum $X_d(f)$ of $x_d(t)$ would be identical, or at least similar, to the spectrum $X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt$ of $x(t)$ at frequencies belonging to the spectral support of $x(t)$. To verify if this condition is satisfied we need to determine the relationship between $X(f)$ and $X_d(f)$. It follows from (2) that

$$X_d(f) = S(f) \ast X(f)$$  \hspace{1cm} (3)

where $\ast$ denotes convolution. To calculate $S(f)$ we note first that $s(t)$ is a periodic function of time and can be represented using Fourier series:

$$s(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi ft/T}$$  \hspace{1cm} (4)
where the coefficients \( c_k \) are given by
\[
 c_k = \frac{1}{T} \int_0^T s(t)e^{-j2\pi k t/T} dt = \frac{1}{T} \sum_{n=1}^{N} e^{-j2\pi n k T/T}
\]  
(5)

Note that \( c_0 = N/T \) and \( |c_k| \leq N/T \). It follows from (4) that
\[
 S(f) = \sum_{k=-\infty}^{\infty} c_k \delta(f - k/T)
\]  
(6)

By combining (3) and (6) we obtain
\[
 X_d(f) = \sum_{k=-\infty}^{\infty} c_k X(f - k/T)
\]  
(7)

The spectrum of the discrete-time signal is therefore a linear combination of shifted and scaled replicas of the spectrum of the original continuous-time signal. In the perfect case we would have \( c_k = 0 \) if \( k \neq 0 \) and
\[
 X_d(f) = \frac{N}{T} X(f)
\]  
(8)

making extracting the signal spectrum from the samples extremely easy. Unfortunately it is impossible to design sampling sequences such that (8) holds for each frequency. This relation could be made true inside the spectral support of \( x(t) \). However, to achieve this effect the sampling frequency must be high, which contradicts our request of maintaining low sampling rates. In [7] we have proposed to tackle the problem in a different way. We considered the case of the signal consisting of one or more narrow-band components whose actual positions are known to be inside a wide range of frequencies. Instead of choosing the sampling sequence so that the suitable coefficients \( c_k \) are zeroed we attempted to minimize their values. In this way we could affect much larger number of coefficients and “clear” a wider range of frequencies of aliasing. For further details how this approach can be used to extract the signal spectrum accurately the reader should refer to [7].

B. Measuring the Level of Aliasing

Now we present a method of measuring the level of aliasing in PNS. This measure is then used for search of optimal sampling sequences. Let \( f_{max} \) be a conservative estimate of the spectral support of the processed signal and \( B_2 << 2f_{max} \) be the actual effective double sided bandwidth of the signal. It follows from (7) that if \( 0 < |k| \leq 2f_{max}T \) then the shifted version of signal spectrum \( X(f - k/T) \) may partially overlap the spectral support and distort relation (8). Therefore we should minimize all \( c_k \) that multiply such spectral replicas in (7).

Our criterion is
\[
 J = \max_{1 < |k| < 2f_{max}T} \left\{ \frac{|c_k|}{c_0} \right\}
\]  
(8)

Let \( k_{max} \) be the largest integer not exceeding \( 2f_{max}T \). Since \( c_0 = N \) and \( |c_k| = |c_{-k}| \), (8) can be simplified to
\[
 J = \frac{T}{N} \max_{0 < |k| < f_{max}} \left\{ |c_k| \right\}
\]  
(9)

\( J \) in (9) can vary between 0 and 1. Zero means that aliasing is fully eliminated while 1 indicates that no suppression of aliasing has been achieved.

III. NUMERICAL AND COMPUTATIONAL ASPECTS OF CALCULATING \( J \)

When designing a PNS sequence we need to take into account certain limitations about how to distribute the sampling instants inside one period. First, it makes sense to assume that all sampling instants are multiples of an elementary period \( L \). This request imposes a limit that the double-sided bandwidth of the spectral support does not exceed \( 1/L \). In our case this means that \( f_{max}L \leq 0.5 \). The second condition takes into account that AD converters need certain amount of time between acquiring two consecutive samples. Hence no two consecutive sampling instants should be closer to each other than some time interval \( H \). We assume that \( H = rL \) and \( T = pL \) where \( r \) and \( p \) are positive integers. For given values of \( r \) and \( p \) one can generate only a finite number of sampling sequences. Hence, in theory, we can find the optimal one by searching the entire space of feasible solutions and comparing their qualities. The risk associated with exhaustive search is that if the space of feasible solutions is large in comparison with the efficiency of the search algorithm and deployed computational power, it may take long time before the result is produced. In [7] we discussed how to reduce the amount of computations necessary to find the optimal solutions. This was achieved by significantly reducing the size of the space of solutions which is subjected to the exhaustive search. In this paper we consider how the computation time can be further reduced by parallelization of the search algorithm and its implementation in a grid environment.

In order to properly explain the details of our grid implementation we present here some details of how the basic optimization algorithm works. This analysis allows us to form some crude estimates on how much
computation is needed to solve the optimization problem and how this time changes when the sizing of the problem is changed.

Criterion (9) uses only the absolute values of the coefficients $c_k$. Therefore by shifting all sampling instants by the same amount of time we do not change the values of $|c_k|$. Without risking that we miss the optimal sequence we can demand that $\tau_1 = 0$. The remaining sampling instants inside the first period are fully characterized by the differences between them and the preceding sampling instant: $\tau_{n+1} = \tau_n + g_n$. Here $g_n$ is the distance (“gap”) between $\tau_n$ and $\tau_{n+1}$. According to our assumptions: $g_n = m_n L \geq H$. The gaps $g_n$, or their integer representations $m_n$, should add up to period $T$. Hence

$$\sum_{n=1}^{N} m_n = p \quad (10)$$

We note that while there is a lower limit on the lengths of the gaps: $m_n \geq r$, there is also an upper one $m_n < 2r$. In fact if we had a gap $m_n = 2r$ or longer than it would be possible to divide it into two shorter ones of length at least $r$ and get an additional sample of the signal. Let $q_1, \cdots, q_{2r-1}$ be the quantities of gaps of size $r, \cdots, 2r-1$ respectively inside one period of the sampling sequence. Then

$$q_r \times r + \cdots + q_{2r-1} \times (2r-1) = p \quad (11)$$

\[ \text{Figure 3. Cost (8) against length of period } T: \ r = 2 \text{ (thin dotted line), } r = 3 \text{ (thin continuous line), } r = 4 \text{ (thick dotted line) and } r = 5 \text{ (thick continuous line)} \]

To generate all possible sampling sequences for given $r$ and $p$ we should find all solutions to equation (11) with respect to $q_r, \cdots, q_{2r-1}$ and then for each of those solutions generate all permutations of gaps. Every such sequence defines uniquely a sampling sequence. The example below illustrates this method. Suppose $r = 3$ and $p = 15$. The permissible gap sizes are 3, 4 and 5 times $L$. Equation (11) takes the following form

$$3q_3 + 4q_4 + 5q_5 = 15.$$  

It has four solutions: 

- $\{q_3 = 5, q_4 = q_5 = 0\}$,
- $\{q_3 = 1, q_4 = 3, q_5 = 0\}$,
- $\{q_3 = 2, q_4 = 1, q_5 = 1\}$ and
- $\{q_3 = 0, q_4 = 0, q_5 = 3\}$.

The number of gap permutations for any of these solutions is given by

$$n_{perm} = \frac{(q_3 + q_4 + q_5)!}{q_3! q_4! q_5!} \quad (12)$$

which gives respectively 1, 4, 12 and 1 sampling sequences per each solution of (11). In this case we have merely 18 different sequences. This number grows quite rapidly when $p$ and $r$ are larger. In a typical case of $p = 100$ and $r = 5$ equation (11) has 520 different solutions while the total number of permutations reaches $3,288,195,610$. In [7] we have shown that in fact only a small proportion of all sampling sequences that can be generated for given $p$ and $r$ needs testing. Despite these savings the amount of computations needed to calculate the magnitudes of coefficients $c_k$ remains large and when the algorithm was implemented on a single PC the waiting times for results could easily reach days. In Figure 3 we illustrate how the cost function changes for different lengths of periods $T$ and $L$ as proportions of $H$.

IV. GRID SOLUTION TO DECREASE EXECUTION TIME

As it was described in section III, the sequential implementation of the algorithm that aims to find the best $J$ value could easily run for days on a single computer. However, the solution could be parallelized relatively easily and run on several computers of a cluster or even distributed on several sites of a Grid. After finding all the solutions for equation (11), different computers can work on different subsets of these solutions generating all permutations of gaps and finding the smallest J value within the specified subset. The overall best solution can then be found as the minimum of the best $J$ values within the subsets. Figure 4 shows how the original algorithm has been parallelised. If the number of solutions for equation (11) is $l$ and the number of parallel branches is $m$, then computer $i$ $(1 \leq i \leq m)$, takes solutions $i, i+m, i+2m, \ldots, i+jm$ until $i+jm \leq l$ and finds the best J value within this subset.

Using this parallelisation two different Grid enabled implementations of the code were generated. In the first implementation the components of figure 4 were taken as GEMLCA [8] legacy codes and a workflow in the P-GRADE Grid portal [9] was created. In the second, a BOINC-based [10] desktop Grid solution was

GRADE portal, and running on the UK National Grid we describe a solution based on GEMLCA and the P-GRADE portal. First, we present a desktop Grid implementation running on laboratory PCs of two UK universities, Westminster and Brunel.

A. Utilising GEMLCA and the P-GRADE Portal

GEMLCA (Grid Execution Management for Legacy Code Applications) [8] represents a general architecture for deploying legacy applications as Grid services without re-engineering the code or even requiring access to the source files. The deployment of a new legacy code service in current GEMLCA implementations assumes that the legacy application is already deployed and runs in its native environment on a Compute Server. The deployment with GEMLCA means to expose this legacy application as a Grid service. It is the task of the GEMLCA Resource layer to present the legacy application as a Grid service to the user, to communicate with the Grid client and to hide the legacy nature of the application. To expose a legacy code as a Grid service with GEMLCA requires only a user-level understanding of the legacy application, i.e., to know what the parameters of the legacy code are and what kind of environment is needed to run the code (e.g., multiprocessor environment with ‘n’ processors). The execution environment and the parameter set for the legacy application is described in an XML-based Legacy Code Interface Description (LCID) file that should be stored in a pre-defined location. This file is used by the GEMLCA Resource layer to handle the legacy application as a Grid service. The LCID file consists of three sections:

- **Environment.** This section contains the name of the legacy code and its binary file; and defines the job manager to be used (Condor, PBS and Fork are supported currently), the maximum number of jobs allowed to be submitted from a single legacy code process, and minimum and maximum number of processors to be used.

- **Description.** This section describes the legacy code in a simple text format.

- **Parameters.** This section exposes the list of parameters, defining for each of them its name, friendly name, type (input or output), order, status (compulsory or optional), file or command line, and the regular expression to be used as input validation.

GEMLCA is integrated with the workflow-oriented P-GRADE Grid portal [9]. The P-GRADE portal enables the graphical development of workflows consisting of various types of executable components (sequential, MPI or PVM programs), execution of these workflows in Globus-based [16] Grids relying on user credentials, and finally the analysis of the correctness and performance of...
applications by the built-in visualization facilities. The P-Grade portal has also been extended with a GEMLCA administration portlet. This portlet hides the syntax and structure of the LCID file from users so that users do not have to know LCID specific details, and do not have to be familiar with possible modifications in legacy code description whenever a new GEMLCA release would require it. The user has to specify exactly the same parameters as in the XML file but this time using a simple Web form. The LCID file is created automatically and uploaded by the portal to the appropriate directory of the GEMLCA resource.

In order to Grid enable the best sampling sequence algorithm, the original program was divided into three different components, as it is illustrated on figure 4. The first component, the pools generator, finds all the solutions of equation (11). These solutions are then fed into compute components that are all working on different subsets of the solutions. These compute components, that are identical to each other just run with different input parameters specifying the actual subset, can work in parallel on different nodes of the Grid. Finally, the output of each worker node is fed into a “best cost” program that finds the solution with minimum J value.

These three program components were written in C and were taken as legacy codes. The executables were deployed on the GEMLCA resources and the LCID files were created using the GEMLCA administration portlet in the P-GRADE portal. This process requires to describe the environmental values of the application and to specify input/output parameter types. Here we use the compute component as an example to illustrate how to publish a legacy code as a GEMLCA Grid service. The process is similar in case of the two other components (pools generator and best cost finder). The Web form and the automatically generated LCID file for the compute program component, is illustrated on figure 5. The job manager specified can either be Condor or PBS depending on the setup of the target cluster. These job managers are responsible to map the execution of the compute program elements to different nodes of a given cluster. The legacy code takes the pool file as input and generates the costs file as the output. An additional input
parameter, the "index", describes the subset of the pool file that the compute component is working on.

Following the deployment of the legacy applications as GEMLCA Grid services, the workflow editor of the portal can be used to create workflows, like it is illustrated on figure 6. In that experiment, 8 workers are working in parallel on different nodes that can be either in one physical cluster or can be distributed to several clusters in the Grid. On figure 6 Pools represents the pools file generator, job 0 to job 7 are the compute components, and BestCost is responsible to select the optimal solution. The figure also illustrates how the jobs, in this example job 3, are mapped onto Grid resources.

After the deployment of the program components as GEMLCA legacy codes, similar workflows can easily be created by the end-users and experiments can be run modifying input parameters like the length of periods $T$ and $L$.

B. Solution Using a BOINC-based Desktop Grid

BOINC (Berkeley Open Infrastructure for Network Computing) [10] is developed by the SETI@home group in order to create an open infrastructure that could be the base for large-scale scientific projects that are attractive for public interest and that can use millions of personal computers for processing their data. This concept enables millions of PC owners to install single software, the BOINC client, and then the PC owners can decide what projects they support with the empty cycles of their computers. There is no need to delete, reinstall and maintain software packages to change among the projects. The properties of BOINC can also be used in smaller scale, combining the power of the computers at institutional level, or even at department level. The SZTAKI Desktop Grid [16] technology is based on BOINC as a well-established open source project, but provides an installation that is specifically tailored for institutional desktop Grids. This modified BOINC installation was used in our experiment to create a desktop Grid version of the digital signal processing application.

The desktop Grid based implementation is also based on the parallelisation model introduced in figure 4 and utilising the fact that the solutions of equation (11) could be processed individually and independently from each other. Each BOINC work unit can include one or more solution vectors of the equation, and can be executed on different worker nodes. The BOINC client application is basically the same compute component that was used in the GEMLCA based solution. The BOINC master, on the other hand, is responsible to create the work packages, distribute them to the workers, collect results and select the solution with the smallest possible $J$ value in the end. BOINC gives low level tools for generic distributed projects to write the necessary code. However, using the DC-API (Distributed Computing API) [11] provided by SZTAKI enables even an application programmer with no Grid and distributed specific knowledge to generate tasks and process results without knowing what grid infrastructure is serving the processing needs.

The architecture of the desktop Grid based implementation and its Website is illustrated on figure 7. The project administrator can set the project parameters like periods $T$ and $L$, and can also specify the number of work units generated. The worker nodes in the current operation consist of more that 100 laboratory PCs (both Linux and Windows machines) at Westminster and Brunel Universities and can be extended basically without any limit. PC donors can download and install the client software from the Website and join the project offering their otherwise not utilised free computing cycles.

![Figure 7. Website and architecture of desktop Grid based DSP solution.](image-url)
V. COMPARISON OF GRID-BASED SOLUTIONS

As the two implemented solutions represent the two main Grid computing approaches, it is interesting and also important to compare how these solutions relate to each other with regards to both usability and performance in order give further guidelines for porting other applications to the Grid.

Once the implementation has been finished the applications had to be handed over for the end-users in order run their experiments. As the end-users are not Grid specialists, easy usability, parameterisation and access to results were top priority for them. In this aspect the GEMLCA – P-GRADE portal solution definitely provides substantial advantages over the Desktop Grid based solution. The portal represents a user friendly interface to create new workflows from previously deployed legacy building blocks, deploy new GEMLCA legacy codes, set parameter values and download results. Moreover, the only client side requirement to use the portal is Internet connection and a Web Browser which allows users to start and analyse experiments wherever they are. On the other hand, the BOINC based solution currently requires more extended knowledge to administer the server and start a new project. BOINC was originally created for projects that sometimes run even for years and thus does not provide a user friendly interface to administer them. Work is currently undergoing to extend the capabilities of the P-GRADE portal and provide support to administer BOINC-based projects in order to overcome this drawback.

In the area of performance the results are two folded. The desktop Grid implementation scales naturally and very large number of workers can be connected to it easily. On the other hand, as neither GEMLCA nor the P-GRADE portal support parameter study like applications currently, the number of parallel branches in the workflow is limited to 15 as the maximum number of input ports to be defined for the Best Cost application. This shortcoming could be overcome by some modification of the Best Cost application, but even in this case the creation of a new workflow with many parallel branches is relatively complicated and time consuming. The new release of GEMLCA and the portal, due in the second half of 2006, will support the definition of parameter ranges and increments, solving this problem.

<table>
<thead>
<tr>
<th>Period T</th>
<th>Sequential</th>
<th>GEMLCA</th>
<th>DGrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>19 min</td>
<td>8 min</td>
<td>1h 17 min</td>
</tr>
<tr>
<td>20</td>
<td>3h 33 min</td>
<td>35 min</td>
<td>1h 44min</td>
</tr>
<tr>
<td>22</td>
<td>41h 53min</td>
<td>7h 23 min</td>
<td>5h 4min</td>
</tr>
<tr>
<td>24</td>
<td>~820 h</td>
<td>~141 h</td>
<td>46h 46min</td>
</tr>
</tbody>
</table>

Table 1. Comparison of GEMLCA and DGrid solutions.

Figure 8. Performance speedup – GEMLCA workflow with 8 parallel branches and DGrid with 100+ computers

Comparing the performance of the solutions and measure their effectives is not straightforward. Where the GEMLCA based solution typically uses dedicated clusters, the desktop Grid solution utilises the free processing power of otherwise busy laboratory PCs that could be different in every single experiment. Instead of a basically impossible and non-realistic precise comparison, we were rather interested in the problem domain where the different solutions could be used. Eight parallel branches of the GEMLCA workflow were specified in the experiments presented in table 1, and compared to a relatively much larger but non-deterministic number of computers (about 120 PCs) used in the desktop Grid experiment. The input parameters were kept fixed except the value of period T. As T increases the sequential execution time increases dramatically. The execution times in the table are rounded to the nearest minute.

The performance results (see figure 8) show that in case of relatively small sequential execution times (meaning small T values), that can be measured in hours rather than days, the GEMLCA based solution is applicable. For example, the performance speed-up in case of defining 8 parallel branches of the workflow is close to a factor of 6. However, the desktop Grid based solution, due to the extensive communication between the master and the worker components and the small size of work units, could produce only very limited speed-up when compared to the sequential solution. Also, in version 4 of BOINC, that was used for this experiment, the master waits for the last work units to be timed out before processing the final result, increasing the execution time significantly. This problem has been overcome in the latest BOINC release. As the execution time increases the desktop Grid solution shows its real power. As scalability is better and easier to realise for the desktop Grid, it is definitely advised to use this solution for T factors over 22.
VI. CONCLUSION AND FURTHER WORK

This paper described a computationally intensive application in order to design a class of optimal periodic nonuniform sampling sequences. The execution of the designed algorithm, even after extensive optimisation, could take days or weeks, making it very difficult for the scientists to run several experiments. Two Grid-based implementations were described and compared in the paper representing the two main Grid computing approaches. The first solution is run on an academic production Grid like the UK National Grid service, and the second could utilise the power of otherwise idle desktop machines. Both solutions are valid and could be used for different experiments with different input parameters.

The presented solutions were designed as a proof of concept and provided a good starting point for the comparison. However, the flexibility and functionality of both solutions are under improvement at the moment. The desktop Grid solution is currently migrated to BOINC version 5 providing better performance, reliability and reducing the latency time after finishing the last work units. Also, the integration of BOINC with the P-GRADE Grid portal is under development and will provide the user friendly interface to run projects and modify parameter values. On the other hand, both GEMLCA and the P-GRADE portal are currently being extended with parameter study support providing better scalability for the production Grid version of the solution.

REFERENCES