Parallel Numerics '05, 81-94 M. Vajteršic, R. Trobec, P. Zinterhof, A. Uhl (Eds.)
Chapter 4: Integration ISBN 961-6303-67-8

Selection of Good Lattice Points Utilizing a Cluster

Bernhard Hechenleitner*, Karl Entacher

Salzburg University of Applied Sciences & Technologies
Schillerstr. 30, 5020 Salzburg, Austria

A Message Passing Interface (MPI) cluster has been used for an extensive parallel search for rank-1 lattice rules using the LLL-Spectral Test [4, 9, 13]. Our parallel search implements the traditional normalization method [6] as well as a new normalization strategy which is proposed in [3]. The resulting lattice parameters of the parallel searches are stored in a database and distributed via a Web-based application server. We shortly introduce the main concepts concerning lattice rules and the spectral test. The concepts for parallelization are explained and some results of exhaustive parallel searches for good lattice points conclude the article.

1 Introduction

The present article shows results of a large scale parallel parameter search for Korobov lattice rules. As a quality criteria for the lattices we used the spectral test with a new normalization strategy. The spectral test allows to perform efficient quality assessments for lattice rules in high dimensions and the parallel setup enables a search over a huge set of parameters. The paper is organized in the following way. In the next sections we shortly introduce the main concepts concerning lattice rules and the spectral test. In Section 2, a detailed description of the parallel approach and the implemented search algorithm is given. Section 3 contains some results of exhaustive parallel searches for good lattice points with regard to different search dimensions using the traditional as well as the new normalization strategy, and Section 4 concludes the article.

*Corresponding author. E-mail: bernhard.hechenleitner@fh-salzburg.ac.at
Research supported by the Austrian Science Fund (FWF) Grand S8311-MAT.

1.1 Good Lattice Points and the Spectral Test

The method of *good lattice points* (GLP) also called *Korobov lattice rules* is a central technique from the fields of *Monte Carlo* (MC) and *quasi-Monte Carlo* (QMC) methods. Good lattice points are classical node sets for QMC integration, defined by the Russian mathematician Korobov [10, 11]. For $y \in \mathbb{R}$ let $\{y\} = y - \lfloor y \rfloor$ be the *fractional part* of y. Consider a vector $\vec{a} \in \mathbb{Z}^s$, $s \geq 2$. A Korobov lattice rule is defined by the set

$$P_m := \{ \vec{x}_n : 0 \leq n < m\}, \text{ with } \vec{x}_n := \left\{ \frac{n \cdot \vec{a}}{m} \right\}, \text{ and modulus } m \in \mathbb{N}. \quad (1)$$

In the following we will use the term Korobov lattice rule P_m only for special vectors $\vec{a}$ defined by a parameter a with $1 < a < m$ and $\vec{a} := (1, a, a^2, \ldots, a^{s-1})$, $s \geq 2$, see [11].

The classical application of Korobov lattice rules is the approximate calculation of integrals over I^s, by the (quasi-) Monte Carlo quadrature rule

$$\int_{I^s} f(\vec{x})\, d\vec{x} \approx \frac{1}{m} \sum_{n=0}^{m-1} f(\vec{x}_n), \quad \vec{x}_n \in P_m. \quad (2)$$

Furthermore, Korobov lattice rules with huge moduli and good lattice quality up to high dimensions provide parameters a and m for linear congruential random number generators (LCGs) with good correlation behavior as a source for MC applications [12, 16]. More recent lattice rules, so-called rank-r lattice rules have been constructed by modular summation over multiples of different vectors $\vec{a}_i$, $1 \leq i \leq r$. Korobov lattice rules are a special case of rank-1 rules. For more details on the theory of integration lattices and their applications in MC and QMC see [16, 18, 17].

The choice of the parameter a heavily determines the distribution quality of the lattice. The central goal for QMC integration is to find lattice parameters a with optimal distribution behavior in different dimensions. For this task, several equidistribution measures for an assessment of the lattice quality have been constructed, see [16, 18, 2, 8].

For our purposes we use the *spectral test*, which can be computed very efficiently and provides a reliable measure for lattice assessment [2]. This test has extensively been applied to find good lattices for several MC and QMC applications, e.g. see [12, 1, 4, 14].

Roughly spoken, the spectral test d_s determines the maximal size of empty slices between hyperplanes of a s-dimensional lattice and therefore measures the distribution quality of such a point set [9, 13]. The spectral test is computed using the Fincke-Pohst algorithm [5]. The magnitude of d_s obviously depends on the dimension and the size of the point set. To enable comparisons

of spectral test results obtained in different dimensions, a normalized spectral test $S_s := d_s^*/d_s$ for which $0 \leq S_s \leq 1$ was introduced [6]. The constants d_s^* are absolute lower bounds on d_s, see [9, p. 105] for $d_s^*, s \leq 8$. Lower bounds for dimension $s > 8$ have been proposed as well in order to compute S_s for arbitrary dimensions [12].

A typical function measuring the "quality" of a lattice parameter a in terms of the spectral test across dimensions is:

$$M_k := \min_{2 \leq s \leq k} S_s \,. \tag{3}$$

Fishman [6] was one of the first who applied this measure to find optimal parameters a for $m = 2^{31} - 1$ in order to get high quality linear congruential random number generators satisfying a fixed threshold $M_6 \geq 0.8$. Recently the measure M_k has been maximized for dimensions up to $k = 32$ in the context of large scale parameter searches [12, 14].

In order to considerably speed up the computations, the LLL basis reduction algorithm [15] may be applied instead of the Fincke-Pohst algorithm as an efficient and reliable approximation to the spectral test. Our experiments use an approximation of this type. The high quality of the LLL-approximation and the speedup with respect to the "original" spectral test have been shown [4].

One problem with the measure M_k (3) is that the magnitudes of the single normalized spectral tests S_s vary significantly for $1 \leq s \leq k$, see [3] for details and examples. In the latter paper a new normalization strategy was suggested in order to make the measure M_k more balanced among the dimensions. Therefore certain regression functions for normalization adjustment were defined:

$$L_s := 0.000042s^3 - 0.0027s^2 + 0.067s - 0.097 \tag{4}$$

$$U_s := -0.000058s^3 + 0.0036s^2 - 0.059s + 1.09 \tag{5}$$

Using these functions the spectral tests S_s were linearly transformed in dimension $2 \leq s \leq 24$ and the transformed measure

$$S'_s = (S_s - L_s)/(U_s - L_s) \tag{6}$$

was used as a new normalized spectral test. For the computer experiments in Section 2 we used both normalizations S_s and S'_s.

2 Parallel Search for GLP

Although the spectral test was chosen as the method for assessing the quality of a lattice, the search for GLPs can become computationally intensive. Therefore, the approach of a parallel application using a cluster was chosen. We restrict our parameter searches to prime moduli m in the range $2^6 < m < 2^{256}$. Parameters for "small" moduli, e.g. $m \leq 2^{31} - 1$ may be applied as lattice rules for QMC-integration, and the parameters for larger m as multipliers for multiplicative LCGs with prime moduli. Therefore our main task is to find the best GLP or multiplier $a \in A$ where A is the set of all primitive roots modulo m since this restriction provides parameters for multiplicative LCGs with full period [16]. As a search criterion we use the measure M_k (3). Concerning the normalization method for the spectral test, the "old" normalization method with S_s and the "new" strategy using S'_s in (3) instead of S_s are distinguished. The basic sketch of the search method is

- Find a primitive root e modulo m. Because m is restricted to be a prime number e is a generator of the cyclic group $\mathbb{Z}_m \backslash \{0\}$ (i.e. the multiplicative order of e modulo m is $\phi(m) = m - 1$ where ϕ is Euler's totient function).

- Take relevant powers $a = e^{\varepsilon} \pmod m$ for which $\gcd(\varepsilon, \phi(m)) = 1$ as multipliers. The latter constraint ensures that a is also a primive root modulo m.

- For each a in the search space and all dimensions up to dimension k, calculate the LLL reduction, find the spectral test value d_s, calculate the defined normalization ("old" or "new"), and find the minimum M_k of the corresponding normalized spectral test values.

- Find the largest M_k value across all chosen multipliers in A.

For this purpose, a prototype of a distributed application for the parallel search of GLPs has been developed.

2.1 Parallel Approach

For increasing efficiency, the PC-cluster *Gaisberg* of the High Performance Computing Group[1] of the department of Scientific Computing at the University of Salzburg has been used to conduct parallel searches. The cluster consists of 25 identically equipped nodes as described in Table 1.

[1] Home page: `http://hpc.sbg.ac.at` (28.09.2004)

Table 1: Cluster nodes.

Architecture	PC
CPU	2 AMD Athlon MP 2800+ (2.1 GHz)
Memory	2 GByte
Operating System	Red Hat Linux 7.3
Linux Kernel	2.4.20
Cluster Interconnect	Scalable Coherent Interface (SCI) from Scali[2]
Programming Interface	MPI from Scali

One master node controls the system setup and the distribution of the search processes among the working nodes. The distributed `Spectral test application` is started at the master node, together with corresponding arguments in the form of command line options. A typical setup is to start two instances of the search application per working node for maximum efficiency, as each node provides two CPUs. The `Spectral test application` at the master node determines important parameters for the search and distributes their values to the search processes at the working nodes utilizing MPI. When a search process finishes its partial search, it passes back its best result – consisting of the best M_k value, its corresponding multiplier a, and the number of executed search loops – to the master process, again by utilizing MPI. The master process sorts the received results and prints out the detailed values. Finally, the best result may be stored in a database.

The basic sequential activities of the `Spectral test application` at the master node as well as at the working nodes is depicted in the Unified Modeling Language (UML) activity diagram in Figure 1. In principle, there are two modes of search operation: exhaustive search and random search. For an exhaustive search, all exponents ε, for which $e^{\varepsilon} \pmod m$ yields a primitive root modulo m, are tested for being the best GLP for the modulus m. For a random search, only a subset of those primitive roots are taken into consideration.

First of all, the MPI initialization and the check of the provided command line options are accomplished. Next, the modulus m and optionally further parameters are set. For getting a rough figure of time consumption, a time measurement is started at all instances of the application. Then, a primitive root e is determined by the master process by means of the following method. First, Euler's totient function $\phi(m)$ of the modulus m is computed. In the case that m is prime, this yields $\phi(m) = m - 1$. Next, the different prime factors $p_1, \ldots, p_k$ of $\phi(m)$ are determined. This is done using a probabilistic integer

[2]Home page: `http://www.scali.com` (28.09.2004)

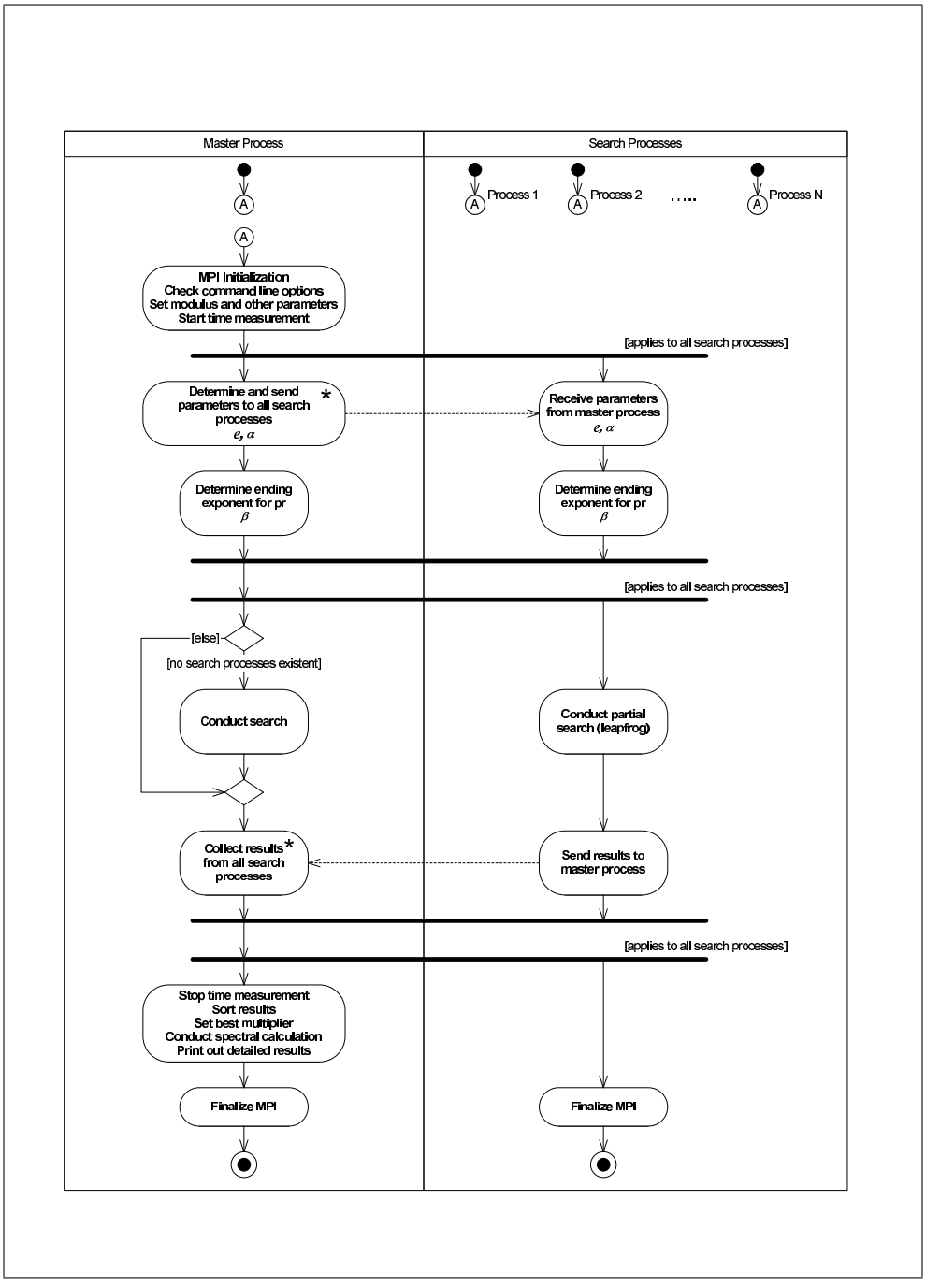

Figure 1: Activity diagram of the distributed `Spectral test application`.

prime-factorization method. Then, for every element e of $\mathbb{Z}_m$ the values[3]

$$e^{\phi(m)/p_i} \pmod{m}, \quad i = 1, \ldots, k \tag{7}$$

are computed. As soon as all of these k results are different from 1, a primitive root e is found. Based on this first primitive root, all further primitive roots $a(\varepsilon, m)$ can be calculated as

$$a(\varepsilon, m) \equiv e^{\varepsilon} \pmod{m} \Longleftrightarrow gcd(\varepsilon, \phi(m)) = 1 \; . \tag{8}$$

The total number of primitive roots for $\mathbb{Z}_m$ is $\phi(\phi(m))$, if m is prime this yields $\phi(m-1)$.

Concerning the search itself, it loops through exponents of e, beginning with the exponent α and ending with the exponent β. For an exhaustive search, α is set to 1, and for a random search, α calculates as the largest ε for which $e^{\varepsilon} \leq m$ plus a random value, depending on the value of m. If the resulting value for α is even, it is incremented by 1. The two values for e and α are distributed to the search processes using MPI. Before being able to search, the upper limit of the exponent β has to be determined by all search processes. For an exhaustive search, this value is simply set to the value of m, for a random search, the doubled number of the wanted search loops is added to α. In the case that no explicit search processes have been started, the whole search may be executed by the master process in a single-threaded approach. However, it is recommended to include several search processes for the sake of division of work. Hence, when using explicit search processes, each of them executes a partial search using a leapfrog method. When a search process has finished its work, it sends back its best result to the master process via MPI. The master process finally stops the global time measurement, sorts the received results, calculates the spectral test values for the best global multiplier and the corresponding values for $M_k, k \in \{8, 16, 24\}$ with regard to both normalization methods, and prints out detailed results. Finally, all processes involved clean up the MPI environment.

2.2 Search Algorithm

The search algorithm for each search process is as follows. Assume, the modulus m, the search dimension k, and the normalization method for the normalized spectral test value *method* have been defined. In the current version of our `Spectral test application` we applied prime numbers $m < 2^{256}$ and dimensions $k \in \{8, 16, 24\}$ and $method \in \{old, new\}$.

[3] e is initialized to 17.

Algorithm 1: Search Loop.

```
WHILE ε < β DO
  IF gcd(ε, m − 1) = 1 THEN
    set multiplier a ⟵ e^ε (mod m);
    set minimum of normalized spectral test value M_k ⟵ 1;
    FOR s ⟵ 2 to k DO
      do LLL reduction;
      find spectral test value d_s;
      set S_s ⟵ d*_s/d_s;
      IF method is new THEN
        set S_s ⟵ (S_s − L_s)/(U_s − L_s);
      ENDIF
      IF S_s < M_k THEN
        M_k ⟵ S_s;
      ENDIF
    ENDFOR
    IF M_k > M*_k THEN
      M*_k ⟵ M_k;
      a* ⟵ a;
    ENDIF
  ENDIF
  ε ⟵ ε + γ;
ENDWHILE
```

In a first step, the values for the best global minimum M_k, denoted as M_k^*, and the best global multiplier a^* are initialized to 0. Then the primitive root e and a starting exponent α, which have been determined and distributed by the master process, are set. Next, the ending exponent β for e, which also represents the loop boundary, is determined and set by each search process. Only those exponents ε are relevant for consideration which fulfill the condition $gcd(\varepsilon, m-1) = 1$. Since m is a prime number, all even values of ε are irrelevant and therefore may additionally be skipped. For this purpose the exponent increment distance δ is set to 2 by default. Applying a leapfrog method across the search processes, the exponent increment value γ for each search process is

$$\gamma = \delta \cdot n_p \tag{9}$$

where n_p is the total number of search processes. The starting exponent ε

for each search process is defined by its rank[4] r_p:

$$\varepsilon = \alpha + \delta \cdot (r_p - 1) \tag{10}$$

After initialization of important search parameters, the search process enters the central search loop, which is described in Algorithm 1. As long as the exponent ε is smaller than the ending exponent β, in a first step it has to be checked if the new exponent yields another primitive root a when applied as $a = e^{\varepsilon} \pmod m$. If not, ε is incremented by γ and the next loop cycle is executed. However, if ε yields another primitive root, a is set as the new multiplier for the modulus m, and the minimum of the normalized spectral test values M_k is initialized to the value 1. For all dimensions $s, 2 \leq s \leq k$, first of all the LLL reduction is calculated and the spectral test value d_s is determined. Next, the normalized spectral test value with regard to the old normalization S_s is calculated. If the new normalization is desired, S_s is transformed to S'_s according to the methods of the new normalization (see Sect. 1.1). If this normalized or transformed value results in a new minimum, M_k is reassigned accordingly. Finally, if the specific multiplier a yields a new highest global value for M_k, then this pair of values (a, M_k) is taken as the new best global pair of values (a^*, M_k^*). Before entering the loop again, the value for ε is increased by γ.

3 Results

First results of parallel searches for GLPs using the parallel setup described in Section 2 are depicted in the following tables. The set of considered moduli consisted of the largest primes smaller than 2^l, for different values of $l \leq 256$.

For the results in Table 2, exhaustive searches for moduli up to $l \leq 31$ have been considered. The search dimension was $k = 16$ with regard to the old normalization. The main intention was to find improved multipliers compared to the tables of L'Ecuyer [12] where exhaustive searches have been performed only for $l \leq 26$. The found improvements are shown as boldface values in the table, together with their corresponding values for $M_k, k \in \{8, 16, 24\}$ for both normalization methods in the first two lines of each entry. For comparison, the previous best multipliers taken from [12] together with their values for M_8 old and M_{16} old (L'Ecuyer used $k \in \{8, 16, 32\}$) are shown as well in line three of each entry. For completeness, their accordant values regarding the new normalization are also depicted in the last line of each entry.

Exhaustive searches for $l \leq 31$ have also been conducted with regard to the new normalization for dimension $k = 24$. The results are shown in Table 3.

[4]The rank of the first search process is 1.

Table 2: Best multipliers a with regard to M_{16} old (exhaustive search).

m	a	M_8 old M_8 new	M_{16} old M_{16} new	M_{24} old M_{24} new
$2^{27}-39$	**66100098**	0.674364	**0.674364**	0.671350
		0.728434	0.728434	0.563524
	3162696	0.702330	0.672640	
		0.751560	0.699585	
$2^{28}-57$	**230195011**	0.692705	**0.680449**	0.680449
		0.734185	0.717931	0.529734
	140853223	0.704620	0.673530	
		0.723500	0.682391	
$2^{29}-3$	**507054386**	0.706185	**0.694721**	0.664672
		0.757498	0.704076	0.545287
	530877178	0.673520	0.670880	
		0.719956	0.593768	
$2^{30}-35$	**790126461**	0.680384	**0.680384**	0.680384
		0.738809	0.726937	0.586649
	295397169	0.683230	0.674200	
		0.749031	0.690916	
$2^{31}-1$	**1257019355**	0.690019	**0.683158**	0.683158
		0.705018	0.705018	0.536430
	784588716	0.658850	0.653880	
		0.689715	0.578044	

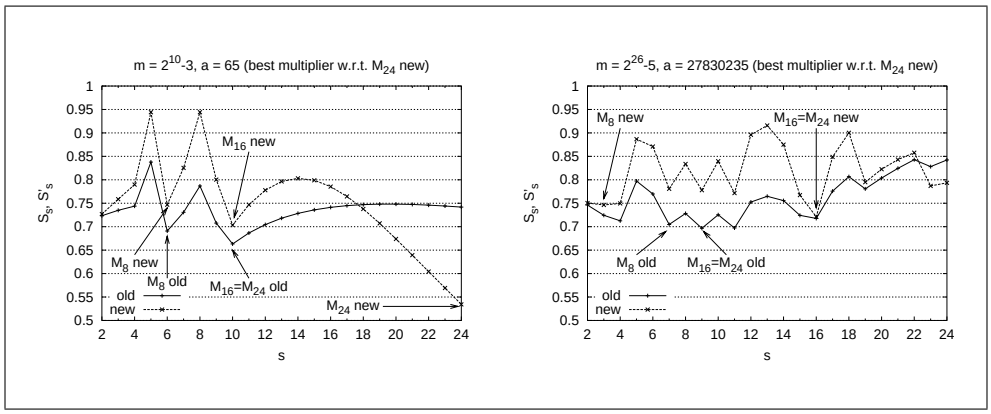

Figure 2: Spectral test behaviors of two different multipliers and moduli.

Figure 2 demonstrates the spectral test behaviors of the best found multipliers for two different moduli. The left graphics shows the behavior of the best multiplier for modulus $m = 2^{10} - 3$. Concerning the old normalization, this multiplier yields a M_{24} old value of 0.663168. Looking at the S'_s values it can be seen that the shape of the curve for higher dimensions strongly decreases because of the intense effects of the normalization adjustments, resulting in a rather low value for M_{24} new of only 0.534339. The special shape of the graph for larger dimensions results from the fact that this modulus and therefore the number of points is very small. Hence for all dimensions $10 \leq s \leq 24$, the corresponding lattices consist of four hyper-planes only resulting in equal non-normalized spectral test values d_s.

Table 3: Best multipliers a with regard to M_{24} new (exhaustive search).

m	a	M_8 new M_8 old	M_{16} new M_{16} old	M_{24} new M_{24} old
$2^{10} - 3$	**65**	0.726822	0.703401	**0.534339**
		0.690694	0.663168	0.663168
$2^{11} - 9$	**1072**	0.597302	0.597302	**0.597302**
		0.599080	0.599080	0.599080
$2^{12} - 3$	**500**	0.665212	0.665212	**0.665212**
		0.642591	0.642591	0.642591
$2^{13} - 1$	**5900**	0.663222	0.663222	**0.663222**
		0.648430	0.648430	0.648430
$2^{14} - 3$	**1543**	0.657914	0.657914	**0.657914**
		0.633986	0.633986	0.633986
$2^{15} - 19$	**7912**	0.726093	0.726093	**0.726093**
		0.673006	0.673006	0.673006
$2^{16} - 15$	**4623**	0.662787	0.662787	**0.662787**
		0.637396	0.637396	0.637396
$2^{17} - 1$	**51308**	0.728530	0.690387	**0.688453**
		0.678984	0.662301	0.662301
$2^{18} - 5$	**152508**	0.686988	0.686988	**0.686988**
		0.663751	0.661700	0.661700
$2^{19} - 1$	**37698**	0.711975	0.711975	**0.707139**
		0.667797	0.667797	0.667797
$2^{20} - 3$	**516672**	0.668804	0.668804	**0.668804**
		0.644146	0.644146	0.644146
$2^{21} - 9$	**1531968**	0.710673	0.710673	**0.710673**
		0.678134	0.674078	0.674078
$2^{22} - 3$	**1135380**	0.686095	0.686095	**0.686095**
		0.672307	0.672307	0.672307
$2^{23} - 15$	**2115063**	0.706835	0.699275	**0.699275**
		0.660804	0.660804	0.660804
$2^{24} - 3$	**926716**	0.696865	0.678597	**0.678597**
		0.655506	0.655506	0.655506

(continued on next page)

Table 3: Best multipliers a with regard to M_{24} new (continued).

m	a	M_8 new M_8 old	M_{16} new M_{16} old	M_{24} new M_{24} old
$2^{25}-39$	**6557845**	0.716337	0.702975	**0.702975**
		0.671214	0.662974	0.662974
$2^{26}-5$	**27830235**	0.746356	0.721056	**0.721056**
		0.704761	0.696578	0.696578
$2^{27}-39$	**33298047**	0.741795	0.732958	**0.719319**
		0.682116	0.682116	0.682116
$2^{28}-57$	**19257650**	0.713572	0.713572	**0.713572**
		0.664384	0.664384	0.664384
$2^{29}-3$	**218346125**	0.713048	0.713048	**0.713048**
		0.669119	0.669119	0.669119
$2^{30}-35$	**353791604**	0.710906	0.710906	**0.710906**
		0.664195	0.664195	0.664195
$2^{31}-1$	**1690867642**	0.700438	0.697860	**0.697860**
		0.661083	0.660640	0.660640

The right graphics in Figure 2 shows the spectral test behaviors of the best multiplier for modulus $m = 2^{26} - 5$. The course of the graph of the S_s values increases for higher dimensions resulting in high values for S'_s as well. Roughly spoken, if an achieved S_s value lies in the lower area of the corresponding spectral test distribution used for the new normalization (cf. Sect. 1.1), then the resulting S'_s value will be adjusted to a value lower than S_s, and if the S_s value lies in the upper area the new normalization results in $S'_s > S_s$. Generally, a parameter search using the new normalization method enables equally stable spectral test behavior with respect to a given threshold across low and high dimensions.

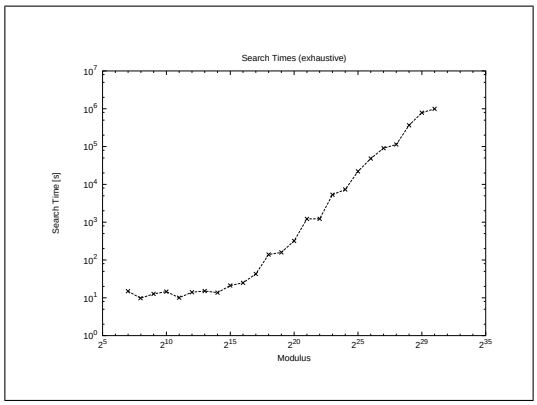

Figure 3: Search times of exhaustive searches with regard to M_{24} new.

Figure 3 shows the time consumption for each search. The shown values include both the serial and the parallel parts of the searches. Note that the serial part is only a minor fraction within a measured time. As the cluster may occasionally also be used by other research groups, the environment was not guaranteed to be unloaded during the searches. The figure shows the general trend that with an increasing value of modulus m the search time increases non-linearly. Therefore exhaustive searches for higher moduli ($l > 31$) have not been conducted.

For selected values of $l \geq 32$ random searches with 5 million search loops for each modulus have been conducted for dimension $k = 24$ with regard to the new normalization. The results of these searches are available electronically at the *Spectral Test Server* [7].

4 Conclusions

We used the spectral test with a new normalization strategy [3] to perform a large scale parallel parameter search for Korobov lattice rules. The new normalization method was chosen to identify parameters with equally stable behavior across low and high dimensions. A selection of the parameters from this experiment is given in the article. The collection of all results of the conducted searches is also available electronically at the *Spectral Test Server* [7]. This Web-based application server offers interactive access to a database, which contains detailed calculation results for many lattice rules, information about scientists working in the field of MC&QMC as well as many publication references. The server further provides the possibility to execute GLP search tasks according to the search algorithm described in Section 2.2 directly via a Web-browser in a single-threaded approach.

Acknowledgments

The authors would like to kindly thank the High Performance Computing Group at the Department of Scientific Computing of the University of Salzburg, Austria, for support and access to their cluster.

References

[1] S.L. Anderson. Random number generators on vector supercomputers and other advanced architectures. *SIAM Rev.*, **32**:221–251, 1990.

[2] K. Entacher, P. Hellekalek, and P. L'Ecuyer. Quasi-Monte Carlo node sets from linear congruential generators. In H. Niederreiter and J. Spanier, editors, *Monte Carlo and Quasi-Monte Carlo Methods 1998*, pages 188–198. Springer, 2000.

[3] K. Entacher, G. Laimer, H. Röck, and A. Uhl. Normalization of the Spectral Test in High Dimensions. Monte Carlo Methods and Applications, **10**(3-4):341–366, 2004.

[4] K. Entacher, Th. Schell, and A. Uhl. Efficient lattice assessment for LCG and GLP parameter searches. *Math. Comp.*, **71**(239):1231–1242, 2001.

[5] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Math. Comp.*, **44**:463–471, 1985.

[6] G.S. Fishman and L.R. Moore. An exhaustive analysis of multiplicative congruential random number generators with modulus $2^{31} - 1$. *SIAM J. Sci. Stat. Comput.*, **7**:24–45, 1986. See erratum, ibid., **7**:1058, 1986.

[7] B. Hechenleitner and K. Entacher. Spectral Test Server, Salzburg Univ. of Applied Sciences and Technologies, Austria. `http://spectral.fh-sbg.ac.at`, 2004.

[8] P. Hellekalek and G. Larcher (eds.). *Random and Quasi-Random Point Sets*, volume **138** of *Lecture Notes in Statistics*. Springer, Berlin, 1998.

[9] D.E. Knuth. *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison-Wesley, Reading, MA, 2nd edition, 1981.

[10] N.M. Korobov. The approximate calculation of multiple integrals. *Dokl. Akad. Nauk SSSR*, **124**:1207–1210, 1959. (in Russian).

[11] N.M. Korobov. Properties and calculation of optimal coefficients. *Dokl. Akad. Nauk SSSR*, **132**:1009–1012, 1960. English transl.: *Soviet Math. Dokl.*, **1**, 696–700.

[12] P. L'Ecuyer. Tables of linear congruential generators of different sizes and good lattice structure. *Math. Comp.*, **68**(225):249–260, 1999.

[13] P. L'Ecuyer and R. Couture. An implementation of the lattice and spectral tests for multiple recursive linear random number generators. *INFORMS Journal on Computing*, **9**(2):209–217, 1997.

[14] C. Lemieux and P. L'Ecuyer. On selection criteria for lattice rules and other quasi-Monte Carlo point sets. *Mathematics and Computers in Simulation*, **55**:139–148, 2001.

[15] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, **261**:515–534, 1982.

[16] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, Philadelphia, 1992.

[17] H. Niederreiter et al. (eds.). Monte Carlo and Quasi-Monte Carlo Methods 1996, 1998, 2000, 2002, 2004. The series of proceedings for the conferences MCQMC 1996 - 2004, Springer Verlag.

[18] I.H. Sloan and S. Joe. *Lattice Methods for Multiple Integration*. Oxford Univ. Press, New York, 1994.