Integer Partitions and the Merit Factor Problem

Blaž Pšeničnik, Borko Bošković, Janez Brest

Computer Architecture and Languages Laboratory, Institute of Computer Science
Faculty of Electrical Engineering and Computer Science
University of Maribor, Koroška c. 46, Maribor, 2000, Slovenia
E-mail: blaz.psenicnik1@um.si, borko.boskovic@um.si, janez.brest@um.si

Abstract.

The merit factor problem is a well-known challenge in combinatorics and signal processing, particularly in designing binary sequences with desirable autocorrelation properties. In this paper, we build upon the approach, which utilizes integer partitions to reduce the search space and prioritize promising regions. We propose a straightforward recursive algorithm for finding integer partitions and a new criterion for identifying and ranking promising search regions. By integrating this criterion with the dual-step GPU optimizer, we improve upon all previously published binary sequences for lengths $400 \leq L \leq 450$.

1 Introduction

In mathematics, an integer partition of a non-negative integer n is expressed as a sum of positive integers, where the order of the summands does not matter. By convention, the number zero has exactly one partition: the empty sum. Since partitions are considered equivalent if they differ only in the ordering of summands, each partition is typically represented as a non-increasing series. For instance, the partition $2 + 1$ can be denoted by the tuple $(2, 1)$. This representation is beneficial for algorithms that exhaustively enumerate all partitions of a given number. An integer partition is restricted if there is a constraint that no partition part is larger than a specified number (and also not smaller, in the case of a doubly restricted partition) [1]. The Ferrers diagrams for all three partitions of $n = 3$ are shown in Figure 1. The total number of partitions of an integer n is given by the partition function $p(n)$, for which no closed-form expression is known. Values of $p(n)$ for $n \leq 49$ can be found in [2].

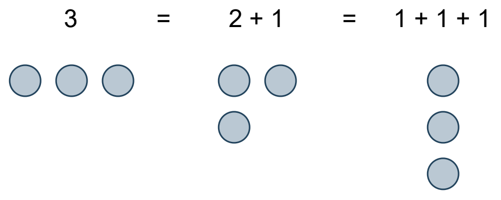

Figure 1: The Ferrers diagrams of the partitions for $n = 3$.

The merit factor problem is a well-known problem in combinatorics and signal processing, particularly in designing binary sequences with desirable autocorrelation properties. Given a binary sequence $S(L) = \{s_1, s_2, \dots, s_L\}$, of length L, where each $s_i \in \{-1, +1\}$, $i = 1, 2, \dots, L$, the aperiodic autocorrelation function C_k at shift k is defined as:

$$C_k = \sum_{i=1}^{L-k} s_i \cdot s_{i+k}, \quad k = 1, 2, \dots, L-1. \tag{1}$$

The merit factor [3] measures the "goodness" of the sequence in terms of how small the sidelobes (non-zero-shift autocorrelations) are. For a given sequence S, the energy of a sequence is defined as $E(S) = \sum_{k=1}^{L-1} C_k^2$. The merit factor is then defined as:

$$F(S) = \frac{L^2}{2 \cdot E(S)}. \tag{2}$$

The optimization goal of the problem is to find a binary sequence S^* that has the maximum possible merit factor $F(S)$ for a given length L and can be formally defined as:

$$S^* = \underset{S \in \{-1,1\}^L}{\arg\max}\, F(S). \tag{3}$$

Despite its simple formulation, the problem is hard and open for general L. It involves a combinatorial search over 2^L possible sequences.

In global navigation satellite systems (GNSS), sequences with low autocorrelation are essential; when combined with additional properties, they are referred to as spreading codes [4]. For example, the GPS L1 C/A signal employs a family of 63 unique spreading codes, each of length 1023 [4]. Such codes are also valuable in low-earth orbit (LEO) satellite applications [5].

In this paper, we build upon the approach of [6], which uses integer partitions to reduce the search space and prioritize promising regions in the merit factor problem. Furthermore, we incorporate the dual-step optimization method from [7] to discover new binary sequences for all lengths $400 \leq L \leq 450$.

The paper is organized as follows. Section 2 reviews the related work. Section 3 presents the proposed algorithm and introduces a new criterion for prioritizing promising search regions. Section 4 discusses the experimental results, and Section 5 concludes the paper.

2 Related work

Several algorithms for generating integer partitions have been proposed in the literature. In [8], the authors introduced an approach for generating restricted integer partitions, leveraging the observation that, in anti-lexicographic order, each partition can be derived from the previous one by decrementing the rightmost part greater than 1 and redistributing the remaining value as early as possible. Any algorithm for generating restricted partitions can naturally be adapted to produce unrestricted ones. In [1], the authors presented an algorithm capable of producing partitions with constant average delay. Average delay is defined as the total time required to generate all partitions divided by the total number of partitions. An algorithm is said to have a constant average delay if this ratio remains bounded by a constant for any given n. This property is desirable in combinatorial generation algorithms because it ensures predictable performance – the time to generate each item does not grow with the size of the input, even if the total number of items does. The comparison of some well-known algorithms is given in [1].

To find optimal binary sequences, the authors in [9] presented an algorithm based on the branch and bound method, which systematically explores the set of all solutions. To reduce the size of the search space, they fixed the m leftmost and rightmost elements of the sequence based on the considered symmetries. The values C_k in Eq. (1) remain unchanged if the sign of each element in the sequence is flipped (multiplied by -1) or if the sequence is reversed. The energy of the sequence remains unchanged in such cases [10]. If every second element of the sequence is complemented, the correlations with odd indices k remain unchanged, while the correlations with even indices only change their sign. Therefore, with the exception of a small number of symmetric sequences, sequences of length 2^N are grouped into eight mutually equivalent classes, called symmetries. The merit factor problem was solved for $L \leq 66$ in [10] with the state-of-the-art algorithm, where the authors estimated that computing the solution for $L = 66$ on a machine with 248 CPU cores would hypothetically take approximately 55 days. This approach quickly becomes intractable, so for longer sequences, stochastic methods that yield good solutions are used [11, 12, 7]. It must be noted that construction methods can be used to construct non-optimal sequences with a certain quality in a very short time. As shown in [13, 14], this approach is suitable for longer sequences with a merit factor value of around 6.3421 or 6.4382.

Due to the exponential growth of the problem's search space, reducing it can be highly beneficial. A skew-symmetric sequence [15], of odd-length $L = 2k + 1$, must satisfy:

$$s_{(k+1)+i} = (-1)^i s_{(k+1)-i}, \quad i = 1, 2, \ldots, k. \tag{4}$$

This constraint reduces the search space to roughly $2^{(L/2)}$ and ensures that $C_k(S) = 0$ for all odd values of k. As a result, the merit factor value increases. However, it is important to note that a skew-symmetric solution is not necessarily optimal for every length. For lengths $L \leq 66$, only 22 optimal sequences are also skew-symmetric [10].

In [6], the authors introduced the use of integer partitions (also called restriction classes) to further reduce the search space to approximately $2^{L/2-n}$ by fixing the first n elements. These elements are selected based on integer partitions of length n, while the subsequent $k - n + 1$ elements are not fixed. The remaining k elements are then set using the skew-symmetry rule, as illustrated in Eq. (5).

For a given value of n, the search can, in theory, be parallelized into $p(n)$ covering subsets, or $2^{n-3} + 2^{\lfloor n/2 \rfloor - 2 + (n \bmod 2)}$ non-covering subsets, by exploiting the aforementioned symmetries [6]. Since exhaustive search is inapplicable for large values of L, the authors proposed potentials and normalized potentials of partitions. These allow the partitions to be ordered and enable the prioritization of promising regions of the search space. The potential of a partition is calculated by partitioning the sequence, setting the center $L - (2 \cdot n)$ elements to 0 (which is the neutral value), and calculating the energy of the resulting sequence. Normalized potential is a modified version of the potential that reduces the weight of shared C_k values of the restriction class, by halving them. Let g be the number of parts (summands) of a partition. In [6], the authors used exhaustive search to compute all potentials and normalized potentials of partitions for $38 \leq n \leq 115$ and selected values of $g \in \{4, \ldots, 12\}$. They also provided a list of sequences with lengths in the range $225 \leq L \leq 527$, achieving merit factor $F \geq 7.0$. Building on this, authors in [7] introduced a parallel GPU implementation and incorporated a second optimization step to enhance the results for longer sequences further.

$$\begin{aligned} S(L) = \ & \underbrace{s_1 s_2 \ldots s_n}_{n} \\ & \underbrace{s_{n+1} s_{n+2} \cdots s_k}_{k-n+1} \\ & \underbrace{s_{k+1} s_{k+2} \cdots s_L}_{L-k} \end{aligned} \tag{5}$$

3 Partitioning the binary sequences

To identify binary sequences with high merit factors, we first need to partition them in order to reduce the search space, as exhaustive search becomes infeasible for large values of L. To achieve this, we require an efficient method for generating all integer partitions of a given number n. For this purpose, we propose Algorithm 1, which is derived from the approaches described in [8, 1]. This straightforward recursive backtracking algorithm exhaustively generates all unrestricted integer partitions of n in anti-lexicographic order, using the standard tuple representation. The algorithm exploits the fact that the next partition can be obtained from the previous one by decrementing the rightmost part greater than 1 and redistributing the remaining value as early as possible [1]. For example, the partition that follows $(4, 3, 1)$ is $(4, 2, 2)$.

The authors in [6] fixed approximately 15% inital elements of the binary sequence using integer partitions.

Algorithm 1 Recursive backtracking algorithm to generate all unrestricted partitions for a given n in anti-lexiographical order.

```
function PARTITIONS(n)
    results ← empty
    function BACKTRACK(left, path, start)
        if length(path) = n then
            if left = 0 then
                append copy of path to results
            end if
            return
        end if
        for $i \leftarrow \min(left, start)$ to 0 step $-1$ do
            append $i$ to path
            BACKTRACK(left $- i$, path, $i$)
            remove last element from path
        end for
    end function
    BACKTRACK(n, empty list, n)
    return results
end function
```

Following their approach, we implemented the proposed algorithm in C++ and evaluated all integer partitions for $40 \leq n \leq 80$, as our target sequence lengths lie in the interval $400 \leq L \leq 450$. We then calculated the potential and normalized potential for each partitioned binary sequence. It is important to note that we did not restrict the values of g in the evaluated partitions; g was allowed to take any value in the interval $\{1, \dots, n\}$.

To verify the correctness of our approach, we computed the number of partitions for selected values of n found by Algorithm 1 and compared the results with the known values of the partition function $p(n)$. Additionally, we validated the computed optimal potentials and normalized optimal potentials against the reference values reported in [6] for various combinations of n and g. As an illustration, Table 1 lists selected partitions that have the minimal normalized potential for given values of n.

Table 1: Partitions with minimal normalized potential for selected values of n.

Partition	n	**Norm. potential**
$11\ 7\ 5^2\ 3^4\ 1$	41	613
$10\ 9\ 5^3\ 3^2\ 2^2$	44	724
$9^2\ 5^3\ 3^6$	51	851

In [7], the authors employed normalized potentials to prioritize regions of the search space; however, this strategy may not yield optimal results. To investigate this, we conducted an experiment in which we executed the self-avoiding walk 100 times from the first step of the dual-step optimization proposed in [7], for $L = 445$. We recorded the average energy of the runs. As shown in Table 2, the partition with the lower optimal normalized potential, which is the one with minimal possible normalized potential for $n = 63$, performed worse on average compared to the alternative partition with a higher normalized potential. This suggests that for larger values of n, the number of parts g may have a greater impact on performance. Notably, the authors in [6] limited g, but this constraint might possibly exclude promising partitions (i.e., search regions). This observation indicates that normalized potential alone may not be a reliable measure of partition quality. As an alternative, we propose ranking partitions based on the average energy of self-avoiding walks on the partitioned sequence. However, since computing self-avoiding walks for every partition is impractical, we instead selected the first few partitions with the lowest potential and normalized potential, for each n, to be evaluated.

Table 2: The average energy of 100 self-avoiding walks for $L = 445$ of a given partition using the suggested parameters in [7].

Partition	g	**Norm. potential**	**Avg. energy**
$11^2\ 5^4\ 3^7$	13	1213	19394
$20\ 11\ 9\ 7\ 6\ 4^2\ 2$	8	2077	19012

4 Results

Using partitions and dual-step optimization [7], we improved all results for skew-symmetric sequences within just a few hours. Since skew-symmetric sequences are of odd length, we applied sequence operators to extend these improvements to even-length sequences as well, subsequently reintroducing them into the second optimization step as suggested in [7]. This approach enabled a comprehensive enhancement across both odd- and even-length sequences.

The improvements we achieved over the previously best-reported merit factors are shown in Figure 2. We improved the merit factor values for all fifty sequences in the interval $400 \leq L \leq 450$, and these binary sequences currently represent the best-known results in the literature.

For five selected values of L, previous best-known merit factor values [6] and new binary sequences are presented in Table 3 by using hexadecimal coding. Each hexadecimal digit is presented with a binary string: $0 = 0000$, $1 = 0001$, ..., $F = 1111$. Therefore, it is necessary to remove the leading 0 values to obtain the correct length of the binary sequence. To get the the sequence values, each 0 of the binary string should be converted to -1. The highest merit factor was found for $L = 433$ reaching 7.8016. This result represents a significant improvement over the previous record of 7.0548 for the same length, as reported in [6].

5 Conclusion

In this paper, we introduced a straightforward recursive backtracking algorithm that enumerates all integer partitions of n in anti-lexicographic order. Through an experiment on $L = 445$ we demonstrated that the lowest nor-

Table 3: New binary sequences for selected lengths L in the range $400 \leq L \leq 450$, along with their merit factor values, compared to the previously best-known merit factor values reported in [6].

L	**Our F**	**Previous F [6]**	**Binary sequence**
405	7.5199	7.0908	1fffffc001ff027b0de1b819e63ec6cfb27696bc93cecdad8fc2d49c1cc4d38d078 76314c6c52659a91a5cb162b55aaad55555
422	7.5981	7.0584	3ffffe0003fe00bc27843c21e32c6b26e4a5ce1ecc99b324f23c99cb72c73399cc e96c87c6e73e4f36974b45a74bd56ab5556aaaaa
433	7.8016	7.0548	1ffffffe0007ff00fc0fc31998de3697849c6c99e30f1e31ec3931a1b392c5b25b4 b2598c6d8e978725c999b2d4ad4ab556aaa5555555
441	7.4823	7.2546	1fffffe000ff83e09f878d3f287b624f92dc6c6613d8313619c6327336326d9a73 b29d3a66c6dc394e27168353c96958a52954aaa555555
450	7.3577	7.0152	1fffffff001ff81f0c38d8f67a44f9338633f09e0cd9a5a164fc4e49cd8e4ed4e7a1 e19cca58b532693394ee16749c92cb5a955aab5555555

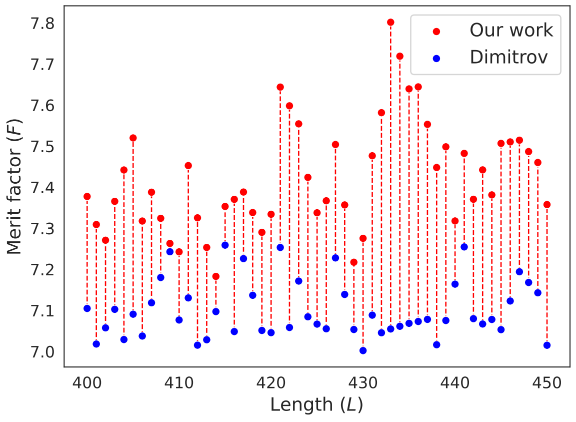

Figure 2: Merit factor values for binary sequences of lengths $400 \leq L \leq 450$ and the previous best-known values reported by Dimitrov [6].

malized potential might not necessary be the best measure to prioritize promising regions the search space. We suggested a new criteria for prioritizing promising search regions (partitions). By combining the new criterion with the dual-step GPU optimizer we improved all previously published binary sequences of all lengths in the interval $400 \leq L \leq 450$.

As the search space grows increasingly complex with larger L, further progress likely demands a different class of optimization methods. Future work will need to explore alternative strategies, such as machine learning-guided search, or hybrid approaches that combine global exploration with local refinement. These more adaptive techniques show promise in navigating the growingly complex search space and pushing merit factors beyond current limits for larger L.

Acknowledgements

The authors (B. B. and J. B.) acknowledge financial support from the Slovenian Research and Innovation Agency (Research Core Funding No. P2-0041 – Computer Systems, Methodologies, and Intelligent Services).

References

[1] Ivan Stojmenović and Antoine Zoghbi. Fast algorithms for genegrating integer partitions. *International journal of computer mathematics*, 70(2):319–332, 1998.

[2] The OEIS Foundation Inc. OEIS A000041: Number of partitions of n. `https://oeis.org/A000041`, 2025. Accessed: 2025-06-25.

[3] Gary L. Mullen and Daniel Panario. Other correlation measures. In *Handbook of Finite Fields*, chapter 10.3.5, pages 322–324. Chapman & Hall/CRC, 1st edition, 2013.

[4] Alan Yang, Tara Mina, Stephen Boyd, and Grace Gao. Large-scale gnss spreading code optimization. In *Proceedings of the 37th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2024)*, pages 948–957, 2024.

[5] Alan Yang, Tara Mina, and Grace Gao. Spreading code optimization for low-earth orbit satellites via mixed-integer convex programming. *EURASIP Journal on Advances in Signal Processing*, 2024(1):67, 2024.

[6] Miroslav Dimitrov. New classes of binary sequences with high merit factor, 2022. arxiv.org/abs/2206.12070.

[7] Blaž Pšeničnik, Rene Mlinarič, Janez Brest, and Borko Bošković. Dual-step optimization for binary sequences with high merit factors. *Digital Signal Processing*, 165:105316, 2025.

[8] JKS McKay. Algorithm 371: Partitions in natural order [a1]. *Communications of the ACM*, 13(1):52, 1970.

[9] Stephan Mertens. Exhaustive search for low-autocorrelation binary sequences. *Journal of Physics A: Mathematical and General*, 29(18):L473, 1996.

[10] Tom Packebusch and Stephan Mertens. Low autocorrelation binary sequences. *Journal of Physics A: Mathematical and Theoretical*, 49(16):165001, 2016.

[11] Borko Bošković, Jana Herzog, and Janez Brest. Parallel self-avoiding walks for a low-autocorrelation binary sequences problem. *Journal of Computational Science*, 77:102260, 2024.

[12] Miroslav Dimitrov. On the skew-symmetric binary sequences and the merit factor problem. *Digital Signal Processing*, 156:104793, 2025.

[13] John Michael Baden. Efficient Optimization of the Merit Factor of Long Binary Sequences. *IEEE Transactions on Information Theory*, 57(12):8084–8094, 2011.

[14] Jonathan Jedwab, Daniel J. Katz, and Kai-Uwe Schmidt. Advances in the merit factor problem for binary sequences. *Journal of Combinatorial Theory, Series A*, 120(4):882–906, 2013.

[15] M. Golay. A class of finite binary sequences with alternate autocorrelation values equal to zero (corresp.). *IEEE Transactions on Information Theory*, 18(3):449–450, 1972.