New variants of SDLS algorithm for LABS problem dedicated to GPGPU architectures

Dominik Żurek, Kamil Piętak, Marcin Pietroń, Marek Kisiel-Dorohinicki

AGH University of Science and Technology
al. Adama Mickiewicza 30, 30-059 Krakow, Poland
{dzurek, kpietak, pietron, doroh}@agh.edu.pl

Abstract. Low autocorrelation binary sequence (LABS) remains an open hard optimisation problem that has many applications. One of the promising directions for solving the problem is designing advanced solvers based on local search heuristics. The paper proposes two new heuristics developed from the steepest-descent local search algorithm (SDLS), implemented on the GPGPU architectures. The introduced algorithms utilise the parallel nature of the GPU and provide an effective method of solving the LABS problem. As a means for comparison, the efficiency between SDSL and the new algorithms is presented, showing that exploring the wider neighbourhood improves the results.

Keywords: LABS, GPGPU, steepest-descent local search

1 Introduction

This paper concentrates on solving the low autocorrelation binary sequence problem using efficient parallel computations on the GPGPU. It introduces a new variant of local search heuristics for LABS together with very efficient realisations designed for the GPGPU architectures. LABS, one of the hard discrete problems despite wide research, remains an open optimisation problem for long sequences. It has wide range of applications including communication engineering [13, 14], statistical mechanics [2, 10] and mathematics [7, 8].

LABS is an NP-hard combinatorial problem with simple a formulation. It consists of finding a binary sequence $S = \{s_0, s_1, \ldots, s_{L-1}\}$ with length L, where $s_i \in \{-1, 1\}$ which minimises the energy function $E(S)$:

$$C_k(S) = \sum_{i=0}^{L-k-1} s_i s_{i+k} \quad \text{and} \quad E(S) = \sum_{k=1}^{L-1} C_k^2(S) \tag{1}$$

The simplest method of solving LABS is exhaustive enumeration that provides the best results, but can be applied only to small values of L. Some researchers use partial enumeration, choosing so-called skew symmetric sequences [11] that are the most likely solutions for many lengths (eg. for $L \in [31, 65]$ 21 best sequences are skew symmetric). Enumerative algorithms are obviously limited to small values of L by the exponential size of the search space. Therefore, a

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_18

lot of various heuristic algorithms have been developed. They use some plausible rules to locate good sequences more quickly. A well-known method for such techniques is *steepest descend local search* (SDLS) [1] or tabu search [6]. In recent years, a few modern solvers based on the self-avoiding walk concept have been proposed. The most promising solvers are *lssOrel* [3] and *xLostavka* [4], which are successfully used for finding skew-symmetric sequences of lengths between 301 and 401 [5]. Another direction of research is using evolutionary multi-agents systems with local optimisation algorithms [9].

In this paper, we propose two new algorithms that are derived from basic SDLS and are implemented on the GPGPU. The first algorithm, SDLS-2, extends the notion of the sequence neighbourhood to a 2-bit distance. The second, SDLS deep through (SDLS-DT), introduces the recurrent exploration of sequences in both the 1-bit and 2-bit neighbourhood. In this paper, we compare them to the SDLS algorithm implemented on the GPGPU described in [15].

2 New variants of SDLS search algorithms for LABS

The new approach to resolve the LABS problem based on SDLS algorithm relies upon increasing the search area of searching during each single iteration.

The implementation of the proposed algorithms utilizes the notion of the neighbourhood of a sequence S with length L obtained by flipping one symbol in the sequence: $N(S) = \{flip(S,i), i \in \{1,..,L\}\}$, where $flip(s_1 \dots s_i \dots s_L, i) = s_1 \dots s_i \dots s_L$ [6].

All computed products can then be stored in a $(L-1) \times (L-1)$ table $T(S)$, such that $T(S)_{ij} = s_j s_{i+j}$ for $j \leq L-i$, and saving the values of the different correlations in a $L-1$ dimensional vector $C(S)$, defined as $C(S)_k = C_k(S)$ for $1 \leq k \leq L-1$. Cotta observed that flipping a single symbol s_i multiples by -1 the value of cells in $T(S)$ where s_i is involved, the fitness of sequence $flip(S,i)$ can be efficiently recomputed in time $O(L)$.

2.1 The SDLS-2 algorithm with extended neighbourhood

The SDLS-2 algorithm extends the notion of neighbourhood to sequences that differ by up to 2 bits. In this case, besides searching for the solution in the neighbourhood with a distance equal to 1, the best results in a single iteration are explored among sequences that differ on two bits with regard to the input sequence. If the best sequence has grater energy than the best sequence from the current iteration, the last one becomes the reference sequence for the next iterations. If the best sequence founded in the current iteration is worse than the input sequence, the algorithm is stopped and the actual reference sequence is returned as a result. In the case of the sequence with the length L, in the single step, this algorithm implies the necessity of looking through $\frac{L(L-1)}{2}$ more solutions than the traditional SDLS algorithm [15].[1]

[1] The GPGPU implementation of discussed approach is not able to check a large space so as a result it provides the worst results. For that reason the detailed description and analysis of this algorithm will be conducted as a future work.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_18

2.2 Sequential version of the SDLS-DT algorithm

In this approach, it is not possible to estimate the number of generated solutions in a single step of the algorithm. The searching of solutions in locality one and two is done in the single step in this case. One step of described algorithm should be defined as a single run of the external loop (Alg. 1, line 3). Inside the body of this loop, the energy after changing the value on the position with the index with the same value as the current loop's counter is calculated first of all (Alg. 1, line 4). L energies, based on the sequences which are different on two bits comparison to the original sequence are then calculated (Alg. 1, line 7). The lowest of the obtained energies is chosen (Alg. 1, line 8) and if its value is lower then the current reference energy (Alg. 1, line 9), update of the reference energy E_r and the input sequence corresponding to it occurs (Alg. 1, line 10). The new run of the external loop then begins. In the following loop runs, there is no possible way to estimate on how many bits the searching space differs comparison to the input sequence because of its dependence upon the winning sequence, which in this case can have a different value on one or two positions. The each single step of the algorithm is done in case of when improvement is not observed (Alg. 1, line 9) (which is equivalent with breaking the *while* loop). The number of that steps is equal to the length of the input sequence. Fig. 1 illustrates two first steps of the algorithm on a sample sequence.

Algorithm 1 Sequential version of SDLS-DT algorithm

```
1:  function SequntialSDLS-DT(S)
2:      E_r = compute_reference_energy(S)
3:      for i := 0 to len(L) do
4:          E[i] = compute_single_energy_by_mutation_i_th_bit(S)
5:          improvement := true
6:          while improvement do
7:              E_local_II = compute_energies_by_mutation_of_two_bits(S)
8:              E_best=compute_lowest_energy(E[i], E_local_II)
9:              if E_r < E_best then improvement := false
10:             update_reference_sequence_and_energy()
            end while
        end for
11:     return E_best
    end function
```

2.3 Parallel version of SDLS-DT for GPGPU

In the GPGPU implementation, the first step is the creation of the external loop with the length L (Alg. 2, line 2). The first operation in this loop changes the value to the opposite value on the position with the index equal to the loop's counter, which is marked as *bit*. The change is realised by thread with that index (Alg. 2, line 3). For that sequence, the structures $C(S)$ and $T(S)$ are created

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_18

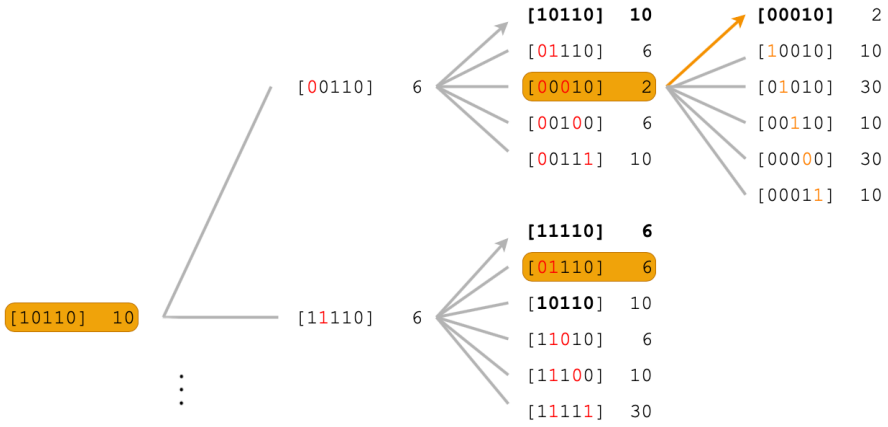


Fig. 1: The example of iteration in SDLS-DT algorithm

(Alg. 2, line 4). Based on the actual sequence, the first energy which provides the first solution in the neighbourhood with distance one is then calculated. This energy becomes the current reference energy (Alg. 2, line 7). The sequence which is different on one position comparison with the original sequence, becomes the input sequence for the second step of the algorithm. The second step of the proposed method is looking for a solution until the next attempts are unable to find a better solution than the current solution which is realised by the internal *while* loop (Alg. 2, line 6). In each iteration of this loop L energies are calculated according to the SDLS algorithm (Alg. 2, lines 7-8). In this case, the thread with ID equal to the counter loop is not blocked because according to this method this thread calculates the energy for the original sequence. From the energies calculated in that way the energy with the lowest value E_{best} is chosen. If this value is lower than the value of the current reference energy E_r (Alg. 2, line 10) it then becomes the reference energy and its sequence becomes the input sequence to the next iteration of the *while* loop (Alg. 2, line 11). In this case, the winning thread actualises the $C(S)$ and $T(S)$ (Alg. 2, line 12). It should be noted that the search could be in a space with a distance higher than two bits. In the second iteration, the input to the internal loop could be the sequence with two different bits than the original sequence. In each internal iteration, the distance between its input sequence and the original sequence could increase. This means that it is not possible to verify which sequence was checked, so there is no sense in blocking any thread. In the case of the occurrence of a *break* in the *while* loop, the second iteration of the external loop begins which changes the bit on the position equal to the counter loop. In the case of the *ith* iteration, the *ith* bit is changed and it becomes the input sequence for the next iteration of the internal loop. In each run of the external loop, the best global energy is actualised in cases in which an in the improvement is observed (Alg. 2, line 13).

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_18

Algorithm 2 Parallel version of SDLS-DT on GPGPU

function PARALLELSDLS-DT(S)
  for $bit := 0$ **to** $len(L)$ **do**
    if $threadId == bit$ **then** $S_{block}[threadId] * = -1$ **end if**
    $create\_T(S)\_and\_C(S)()$
    $E_r := compute\_energy\_with\_local\_one(S_{block})$
    while $improvement$ **do**
      $mutation\_of\_threadId\_bit(S_{block})$
      $ParallelValueFlip(S_{block}, T', C')$
      $E_{best} := compute\_lowest\_energy()$
      if $E_r < E_{best}$ **then** $improvement := false$ **end if**
      $update\_reference\_sequence\_and\_energy()$
      $update\_T(S)\_and\_C(S)()$
    end while
    if $E_{best_{global}} < E_{best}$ **then** $E_{best_{global}} := E_{best}$ **end if**
  end for
  return $E_{best_{global}}$
end function

Sequence length	Method	Number of searched solutions	Average	Standard deviation	Number of running kernels
128	**SDLS**	20 740 434 073	2807	531.8	57773
128	**SDLS-2**	16 865 830 144	224 089	41837.6	593
128	**SDLS-DT**	31 176 794 592	356 752	46302.4	703
256	**SDLS**	22 380 865 408	11159	2 836	15 661
256	**SDLS-2**	15 975 437 994	1 657 780	257 819	74
256	**SDLS-DT**	24 978 089 130	3 019 130	292 121	68

Table 1: The number of explored solutions during one minute computations

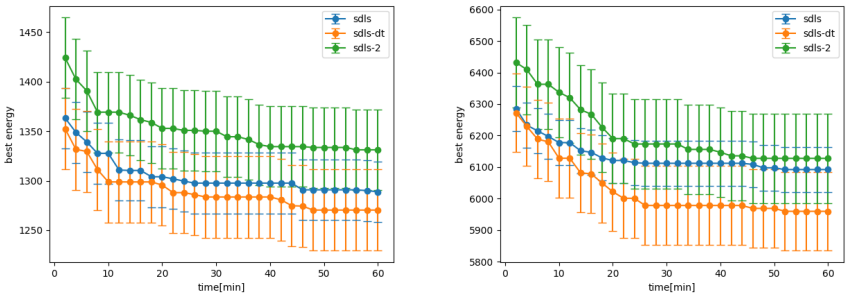
3 Effectiveness of the proposed algorithms

In order to measure the effectiveness of the proposed algorithms, each parallel version of them was performed on the Nvidia Tesla V100-SXM2-32GB[2]. Each algorithm was seeking the optimal solution for three different input lengths $(128, 256)$. In the first iteration, the processor randomises 128 different sequences, one for each GPU block. With data generated in such a way, each kernel starts searching the minimum value of energy. The moment that the best energy from each block is found, it is stored as the current best energy $E_{global\_optimum}$. The processor generates a new set of 128 sequences for which the algorithm repeats the search process and if applicable, the global energy is updated. The entire process was run 10 times for one hour each.

Table 1 contains the number of searched solutions along with the average of searched solutions for the single thread block and the number of individual kernels that are run. Fig. 2 presents the efficiency of the algorithms. As could

[2] https://www.nvidia.com/en-us/data-center/v100/

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_18



(a) Energy of LABS $L = 128$ (b) Energy of LABS $L = 256$

Fig. 2: Energies achieved by basic SDLS, SDLS-2 and SDLS-DT algorithms

be observed for each size of the problem, the best solution was obtained by the *SDLS-DT* algorithm. This fact is due to the parallel implementation of this algorithm was able to successively explore significantly larger space than SDLS and SDLS-2 algorithms. Despite the results being similar to those achieved with SDLS, the efficiency of the SDLS-2 algorithm is weaker. This fact proves that the efficiency of the algorithm is high but the proposed implementation is not effective. The improvement of this implementation is considered as future work.

4 Conclusions and further work

This paper is the next step of our research related to efficient algorithms for LABS realised using GPGPU architectures. The presented new SDLS algorithms show a significant improvement in effectiveness compared to the traditional SDLS approach. They can be further combined with meta-heuristics such as evolutionary algorithms, which constitute a basis for the concept of a hybrid environment in the master-slave model that was proposed in [12].

In the near future the authors plan to implement parallel variants of the self-avoiding walk, *lssOrel* or *xLostavka* solvers. We also consider extending the search neighbourhood in tabu search heuristics and propose new variants of the algorithm dedicated to the GPGPU. The next interesting and promising direction of research is to design and implement an evolutionary multi-agent system with new local optimisation on the GPGPU units.

Acknowledgments The research presented in this paper was realized thanks to funds of Polish Ministry of Science and Higher Education assigned to AGH University of Science and Technology.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_18

Bibliography

[1] Bartholomew-Biggs, M.: The Steepest Descent Method, pp. 1–8. Springer US, Boston, MA (2008)

[2] Bernasconi, J.: Low autocorrelation binary sequences : statistical mechanics and configuration space analysis. Journal De Physique **48**, 559–567 (1987)

[3] Bošković, B., Brglez, F., Brest, J.: Low-Autocorrelation Binary Sequences: On Improved Merit Factors and Runtime Predictions to Achieve Them. arXiv e-prints arXiv:1406.5301 (Jun 2014)

[4] Brest, J., Bošković, B.: A heuristic algorithm for a low autocorrelation binary sequence problem with odd length and high merit factor. IEEE Access **6**, 4127–4134 (2018). https://doi.org/10.1109/ACCESS.2018.2789916

[5] Brest, J., Bošković, B.: In searching of long skew-symmetric binary sequences with high merit factors (2020)

[6] Gallardo, J.E., Cotta, C., Fernández, A.J.: Finding low autocorrelation binary sequences with memetic algorithms. Appl. Soft Comput. **9**(4), 1252–1262 (Sep 2009)

[7] Günther, C., Schmidt, K.U.: Merit factors of polynomials derived from difference sets (2016)

[8] Jedwab, J., Katz, D.J., Schmidt, K.U.: Advances in the merit factor problem for binary sequences. Journal of Combinatorial Theory, Series A **120**(4), 882 – 906 (2013)

[9] Kowol, M., Byrski, A., Kisiel-Dorohinicki, M.: Agent-based evolutionary computing for difficult discrete problems. Procedia Computer Science **29**, 1039 – 1047 (2014)

[10] Leukhin, A.N., Potekhin, E.N.: A bernasconi model for constructing ground-state spin systems and optimal binary sequences. Journal of Physics: Conference Series **613**, 012006 (may 2015)

[11] Packebusch, T., Mertens, S.: Low autocorrelation binary sequences. Journal of Physics A: Mathematical and Theoretical **49**(16), 165001 (Mar 2016)

[12] Piętak, K., Żurek, D., Pietroń, M., Dymara, A., Kisiel-Dorohinicki, M.: Striving for performance of discrete optimisation via memetic agent-based systems in a hybrid cpu/gpu environment. Journal of Computational Science **31**, 151 – 162 (2019)

[13] Zeng, F., He, X., Zhang, Z., Xuan, G., Peng, Y., Yan, L.: Optimal and z-optimal type-ii odd-length binary z-complementary pairs. IEEE Communications Letters **24**(6), 1163–1167 (June 2020)

[14] Zhao, L., Song, J., Babu, P., Palomar, D.P.: A unified framework for low autocorrelation sequence design via majorization–minimization. IEEE Transactions on Signal Processing **65**(2), 438–453 (Jan 2017)

[15] Żurek, D., Piętak, K., Pietroń, M., Kisiel-Dorohinicki, M.: Toward hybrid platform for evolutionary computations of hard discrete problems. Procedia Computer Science **108**, 877 – 886 (2017), international Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_18