EFFECTIVE ENERGY RECOMPUTATION FOR LOW AUTOCORRELATION BINARY SEQUENCE PROBLEM

Leonid Hulianytskyi, Vladyslav Sokol

Abstract:*This paper deals with finding low autocorrelation binary sequences which is a hard combinatorial optimization problem. Recent developments in this area analyzed, in order to understand characteristics of a problem. Emphasis is put on effective energy recomputation operators. Different types of these operators are tested to achieve full picture of LABS solvers development process. It is shown that latest state-of-the-art metaheuristics in fact all based on simplest Tabu Search framework, achieving performance boost by means of energy recomputation operators' optimization. In this paper we construct variation of memetic algorithmincorporating latest developments to reach higher performance than it's original. Comparison to a state-of-the-art TSv7 approach completed on instances with known optimums as well as on some unsolved larger ones. It is concluded that these approaches shows similar performance as they both have the same built-in heuristic. As further research proposed a comparison of different metaheuristic frameworks applied to this problem.*

Keywords: *combinatorial optimization, low autocorrelation binary sequences, memetic algorithm,stochastic local search, tabu search.*

ACM Classification Keywords: *G. 1. 6. Mathematics of Computing, Numerical Analysis, Optimization.*

Introduction

Findinglow autocorrelation binary sequence (LABS)is a hard combinatorial optimization problem withmany applications in different areas such as telecommunications, physics and chemistry (see Ref.[Gallardo, Cotta, Fernandez, 2009]). It has been deeply studied since the 1960sby both the communities of Physics and Artificial Intelligence.

Binary sequence $S$of length Lrepresented as$s_1 s_2 \dots s_L$, with $s_i \in \{-1,1\}$for $1 \le i \le L$, i.e.,$S \in \{-1,1\}^L$. The aperiodic autocorrelation of elements in sequence $S$with distance $k$is defined as:

$$C_k(S) = \sum_{i=1}^{L-k} s_i s_{i+k}. \quad (1)$$

The energy function associated with sequence S is the quadratic sum of its correlations:

$$E(S) = \sum_{k=1}^{L-1} C_k^2(S). \quad (2)$$

The LABS problem with length L,or$LABS(L)$, liesin finding a binary sequence of length L with associated minimum energy.

Related work

For the last 3 decades the LABS problem has been constantly tackled in the literature using exact and heuristic methods (Ref. [Gallardo, Cotta, Fernandez, 2009]).

In [Golay, 1982]for the first time, were published optimal solutions for$L \in [6,32]$ that author computed by performing an exhaustive search enumeration.

Mertens in[Mertens, 1996]used a parallel branch and bound algorithm with symmetry breakingprocedures to solve instances up to $L = 60$.

Presented in [Dotu, van Hentenryck, 2006]stochastic local search (SLS) - Tabu Search (TS) algorithm, was capable of reaching global optimums (GO) for$6 \leq L \leq 48$ faster than exact enumeration approaches.

Gallardo et al.([Gallardo, Cotta, Fernandez, 2007],[Gallardo, Cotta, Fernandez, 2009]) developed Memetic Algorithm (MA) endowed with TS mechanism inspired by [Dotu, van Hentenryck, 2006]. Results shows that proposed algorithm worked as state-of-the-art at time, finding optimal solutions for instances with known optima faster than predecessors overall by at least an order.

In[Halim, Yap, Halim, 2008] authors adopt and significantly improve TS framework of [Dotu, van Hentenryck, 2006] so it performs actually even better than recent state-of-the-art MA. According to authors, such major improvements became available after deep analysis of search trajectory, and as a result more effective diversification part of their metaheuristic.

In this paper we will try to understand nature of such performance boost in both algorithms of MA and TSv7. Moreover we will use gathered data to construct effective framework that outperforms previous approaches.

Memetic Algorithm

MA presented in [Gallardo, Cotta, Fernandez, 2007], [Gallardo, Cotta, Fernandez, 2009] is a metaheuristic consisting of GeneticAlgorithm and TSbuilt inside it. TS in this approach is quite similar to one proposed earlier in [Dotu, van Hentenryck, 2006] and have basic structure as one can find in simplestTS implementation. At each step algorithm moves to the best (in terms of energy) solution in neighborhood, even if its energy is higher, using a short-term memory (tabu list) to avoid cycling.

Authors of MA performed calibration of framework parameters to hit better performance, but key improvement of their approach was lying in effective energy recomputation at each step. Instead of naive full energy recalculation (as in (2), $O(L^2)$) every time algorithm wanted to estimate move to another solution, authors proposed more efficient incremental operator ($O(L)$). Tradeoff asalways in such cases in memory (which we actually have enough): solution besides actual sequence and energy value contains two additional data structures: matrix $T \in (L-1) \times (L-1)$, and vector $C \in (L-1)$. $T(S)$stores all computed products such that $T(S)_{ij} = s_j s_{j+i}$for $j \leq L - i$, when $C(S)$containsvalues of the different correlationsC_k (as in (1))(Figure1).

	1	2	3	4	
1	s_1s_2	s_2s_3	s_3s_4	s_4s_5	$s_1s_2 + s_2s_3 + s_3s_4 + s_4s_5$
2	s_1s_3	s_2s_4	s_3s_4		$s_1s_3 + s_2s_4 + s_3s_4$
3	s_1s_4	s_2s_5			$s_1s_4 + s_2s_5$
4	s_1s_5				s_1s_5

Figure 1.Example of additional data structures $T(S)$and $C(S)$ for $L = 5$ instance.

Regarding this, an efficient operator for computing energy of solution, 1 bit different from current, can be used (Figure2).

```
        FlipedEnergy(L, T, C, i)
        {
flipedEnergy = 0
fork = 1toL - 1 do
            {
                c = C[k]
if (i - k>= 1)
                    c = c - 2 * matrix[k, i –k]
if (i + k<=L)
                    c = c - 2 * matrix[k, i]
          flipedEnergy = flipedEnergy + c * c
            }
return flipedEnergy
        }
```

Figure 2.Pseudocode of an efficient energy recomputation operator.

After selecting best move in neighborhood algorithm as well updates additional data structures using similar $O(L)$-performance operator (Figure 3).

```
        Update(L, T, C, i)
        {
fork = 1toL - 1do
            {
if (i - k>= 1)
                {
C[k] = C[k] - 2 * T[k, i - k]
T[k, i - k] = -T[k, i - k]
                }
if (i + k<= L)
                {
C[k] = C[k] - 2 * T[k, i]
T[k, i] = -T[k, i]
                }
            }
        }
```

Figure 3.Pseudocode of a selected move update operator.

Data structures $T(S)$ and $C(S)$ initialized once, at the beginning of each TS procedure.

We conducted several experiments to see impact of such energy recalculation on overall performance of this metaheuristic. Here and further experiments were performed on instances of sizes $L \in [31,60]$ for which global optimums (GO) are known (http://www-e.uni-magdeburg.de/mertens/research/labs/open.dat), on a Core i7-860 2.8 GHz PC, programmed in C# (source code available upon email request).Termination criteria for each algorithm set as finding a GO. Results are - timings in seconds, showing how long it takes to reach GO. Each experiment was repeated 10 times for statistical estimation.

Table 1 and Figure 4 show results obtained by Memetic Algorithm with full energy recomputation (MAf) and original MA with effective incremental energy recomputation.

Table 1. Average performance of MAf, MA in seconds for each instance size L.

L	MAf	MA	L	MAf	MA	L	MAf	MA
31	0.33	0.07	**41**	74.38	15.33	**51**	3889.45	478.4
32	1.17	0.19	**42**	74.86	5.34	**52**	1602.03	241.28
33	1.59	0.52	**43**	429.34	35.55	**53**	1998.13	108.38
34	10.3	0.36	**44**	117.05	13.92	**54**	8536.41	234.62
35	18.27	1.97	**45**	73.79	23.24	**55**	4738.36	600.05
36	6.61	0.89	**46**	110.5	9.72	**56**	5207.26	901.57
37	12.25	1.17	**47**	110.17	12.49	**57**	25598.61	1235.28
38	17.16	1.75	**48**	650.68	44.76	**58**	26376.22	1231.7
39	67.83	5.64	**49**	468.51	34.22	**59**	12105.19	966.71
40	32.17	3.95	**50**	204.1	41.21	**60**	15240.15	1813.36

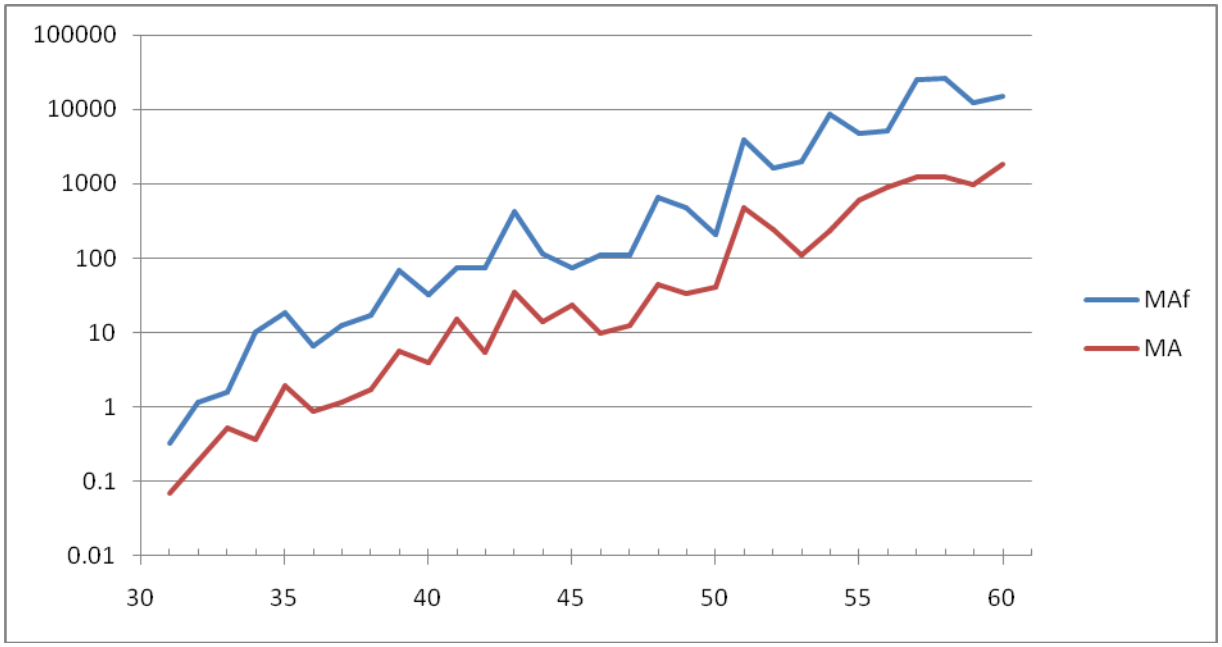

Figure 4. Average performance of MAf, MA in seconds for each instance size L, logarithmic scale.

As we can see changing only the energy recomputation results in a major speed boost of about 10 times in average for this set of instances. This result confirms obvious conclusion that energy calculation takes most of CPU time used by heuristics.

Tabu Search

In [Halim, Yap, Halim, 2008] proposed TS similar to [Dotu, van Hentenryck, 2006] with some major improvements (called TSv7).First of all,an efficient incremental recomputation of energy, inspired by MA were used. Next, according to authors such simple diversification as random restarts used in [Dotu, van Hentenryck, 2006]was not efficient due to some features of a search space. As we can see binary sequence have the same energy as negated or inversed one, making each instance of problem to have at least 4 GO. Authors of [Halim, Yap, Halim, 2008] provided some empirical evidence, achieved with exact enumeration for smaller instances, that current solution of TS is always much closer(in terms of Hamming distance) to one GO than to others. Thus, it was suggested to make small local restarts instead of full random reconfiguration, each time heuristic completes intensification phase. Experiments showed that such approach could find optimal solutions approximately 2 times faster than previous state-of-the-art MA. Note that original TS realization in [Halim, Yap, Halim, 2008] called TSv1 actually showed similar effectiveness as TSv7, outperforming MA in about an order of 2 in average (yet slightly inferior comparing to TSv7).

When programming TSv7 approach on our platform we examine original source code (thanks to authors for sharing it). It is come to our attention that energy recomputation used in this approach is slightly different from the way proposed in MA. Authors put into use one more additional data structure $T2(S) \in (L) \times (L)$ that stores some temporary calculations used in FlipedEnergy() operator. Such manipulations helped to optimize FlipedEnergy() operator by transferring some computational complexity to another operator Update(). Note that Tabu Search approach on each step uses approximately L times FlipedEnergy() operator and only once Update() operator when selected best solution from neighborhood, thus making this optimization useful. Initialization of $T2(S)$ conducted once after every reconfiguration of sequence (Figure 5).

```
for i = 0toL -1 do
            {
k = 1
do
                {
T2[k, i] = 0
if (i -k>= 1)
T2[k, i] = T2[k, i] + 2 * T2[k, i - k]
if (i + k<=L)
T2[k, i] = T2[k, i] + 2 * T2[k, i]
k = k + 1
                }
whileNOT(i - k< 0 ANDi + k >= L)
            }
```

Figure5.Pseudocode of $T2(S)$ structure initialization.

FlipedEnergy() and Update() operators then can be successfully changed to achieve maximal performance of algorithm (Figure 6, Figure 7).

```
        FlipedEnergy(L, T2, C, i)
        {
flipedEnergy = 0
fork = 1toL- 1do
            {
                c = C[k] - T2[k, i]
                flipedEnergy = flipedEnergy + c * c
            }
return flipedEnergy
        }
```

Figure 6. Pseudocode of an efficient energy recomputation operator, optimized with usage of an additional data structure $T2(S)$, instead of $T(S)$.

```
        Update(L, T, T2, C, i)
        {
forkk = 1to L - 1do
            {
C[k] = C[k] - T2[k, i]
if (i - k>= 1)
                {
T2[k, i] = T2[k, i] - 4 * T[k, i - k]
T2[k, i - lag] = T2[k, i - k] - 4 * T[k, i - k]
T[k, i - lag] = -T[k, i - k]
                }
if (i + k<= L)
                {
T2[k, i] -= 4 * T[k, i]
T2[k, i + k] -= 4 * T[k, i]
T[k, i] = -T[k, i]
                }
            }
        }
```

Figure 7. Pseudocode of update operator for data structures after successful move selection.

We conducted some experiments to show importance of such optimization. For this we programmed 3 versions of TSv7 respectively: with full energy recomputation on each step (TSv7f), with efficient incremental energy recomputation (TSv7e) as in MA, and original with effective optimized energy recomputationTSv7 (Table 2).

Table 2. Average performance of TSv7f, TSv7e,TSv7 in seconds for each instance size L.

L	TSv7f	TSv7e	TSv7	L	TSv7f	TSv7e	TSv7	L	TSv7f	TSv7e	TSv7
31	0.25	0.06	0.04	**41**	113.85	10.05	5.05	**51**	3110.59	550.44	147.01
32	1.06	0.1	0.04	**42**	94.06	8.44	4.81	**52**	5172.52	408.45	245.52
33	5.22	0.37	0.16	**43**	614.64	44.48	39.48	**53**	4147.09	223.11	163.5
34	2.41	0.42	0.15	**44**	120.75	7.82	8.07	**54**	3028.22	169.24	102.18
35	5.77	0.62	0.56	**45**	292.2	11.37	5.21	**55**	5122.86	370.79	173.83
36	5.5	0.39	0.48	**46**	26.88	9.03	3.15	**56**	12113.13	523.82	656.46
37	18.87	0.98	0.64	**47**	128.52	17.72	5.35	**57**	26773.11	1817.93	746.44

38	20.72	1.31	1.47	**48**	487.03	39.84	14.95	**58**	21412.45	1853.58	890.86
39	51.5	4.14	2.8	**49**	402.59	43.98	26.9	**59**	3289.51	1021.54	632.7
40	55.96	2.97	1.85	**50**	313.61	20.29	19.69	**60**	4535.86	1535.78	753.22

Figure5 shows performance of both sets of algorithms (MAs' and TSv7s').

Figure 5. Average performance of MAf, MA,TSv7f, TSv7e,TSv7 in seconds for each instance size L, logarithmic scale.

We can split algorithms into 3 visual groups by performance. The weakest two approaches are both algorithms with full energy recalculation showing approximately same average time to find GO. The next group contains of MA and TSv7e both of which got efficient recomputation of energy. These approaches are as expected much more efficient then the first two, and again showing same performance. Last algorithm - TSv7 indeed outperforms all opponents, having more optimized effective recomputation comparing to MA and TSv7e.

Optimized Memetic Algorithm

Considering results obtained above we decided to improve performance of MA approach (we will call it OMA). We added optimized energy recomputation as in TSv7. Then we reconfigured some parameters aiming to a better performance. GA used in MA was under next parameter settings: population size – 100, crossover rate 0.9, mutation rate – 1/L, binary tournament selection, uniform crossover. After empirical adjustments we decided to change mutation rate to 0.01 so mutation frequency does not degrade with increase of L. Next step was reconfiguration of embedded TS heuristic. We changed iterations parameter from 3*l/2 in MA to 4*L in our OMA, as experiments shows that it is worth to spend additional time to intensification. Tabu tenure parameters set as following: minimal tabu tenure set as 1+L/20, random additional tabu tenure set as L/10. Exact same parameter values were used in TSv7 which is another empirical evidence of effectiveness of such calibration. Table 3 shows results of experiments on instances with known optima.

Table 3. Average performance of OMA in seconds for each instance size L.

L	OMA	L	OMA	L	OMA
31	0.04	**41**	5.09	**51**	314.26
32	0.2	**42**	5.51	**52**	139.15
33	0.22	**43**	11.19	**53**	67.38
34	0.3	**44**	6.86	**54**	72.03
35	0.38	**45**	5.22	**55**	169.76
36	0.38	**46**	2.43	**56**	471.47
37	1.13	**47**	6.61	**57**	534.22
38	0.59	**48**	25.79	**58**	952.94
39	3.64	**49**	16.5	**59**	634.33
40	1.38	**50**	32.5	**60**	638.54

Figure6 shows performance ofOMA in comparison to original TSv7 which is current state-of-the-art approach.OMA performance is comparable to TSv7 but slightly better in average.

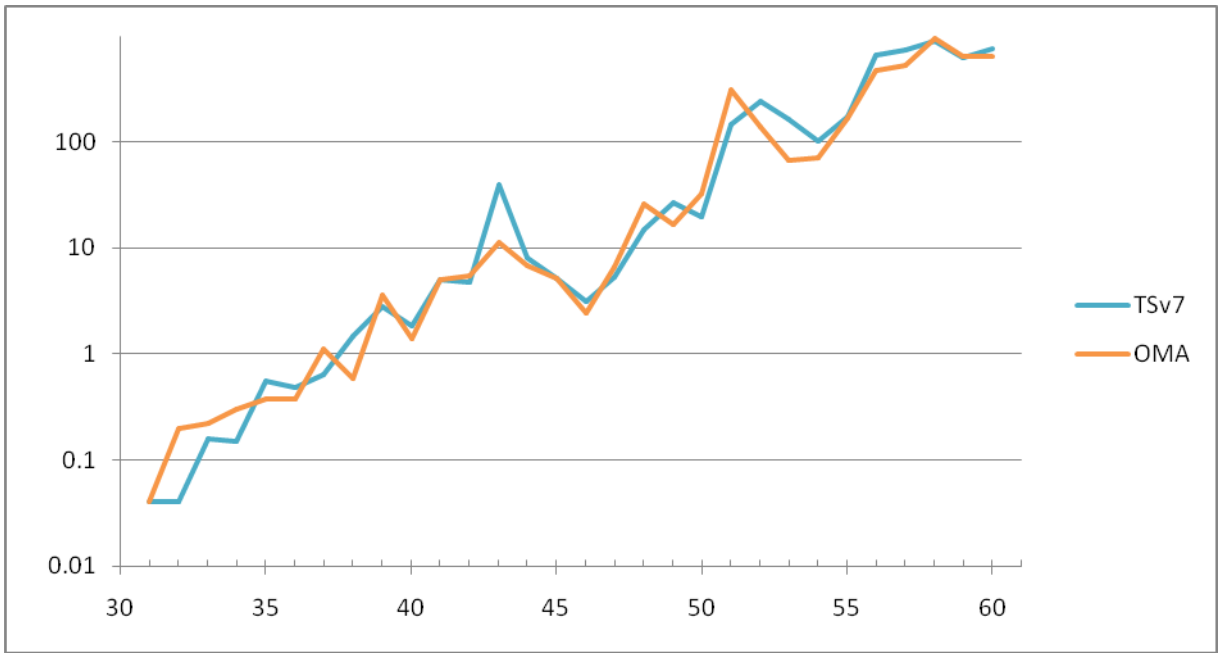

Figure 6. Average performance of TSv7, OMA in seconds for each instance size $\mathbb{L}$, logarithmic scale.

Experiments on larger instances

Another important part of OMA and TSv7 effectiveness comparison would be experiments on instances for which GO are not found yet or not proven. For this analysis we selected 10 next instances $\mathbb{L} \in [61,70]$. Termination criteria for both approaches was set as time limit and defined as follows. Previous experiment showed us average time needed by metaheuristics to reach GO at specified $\mathbb{L}$. We can approximate time with linear trend and make forecast to achieve time limit formula we need. For this purpose we used results for instances $\mathbb{L} \in [51,60]$.

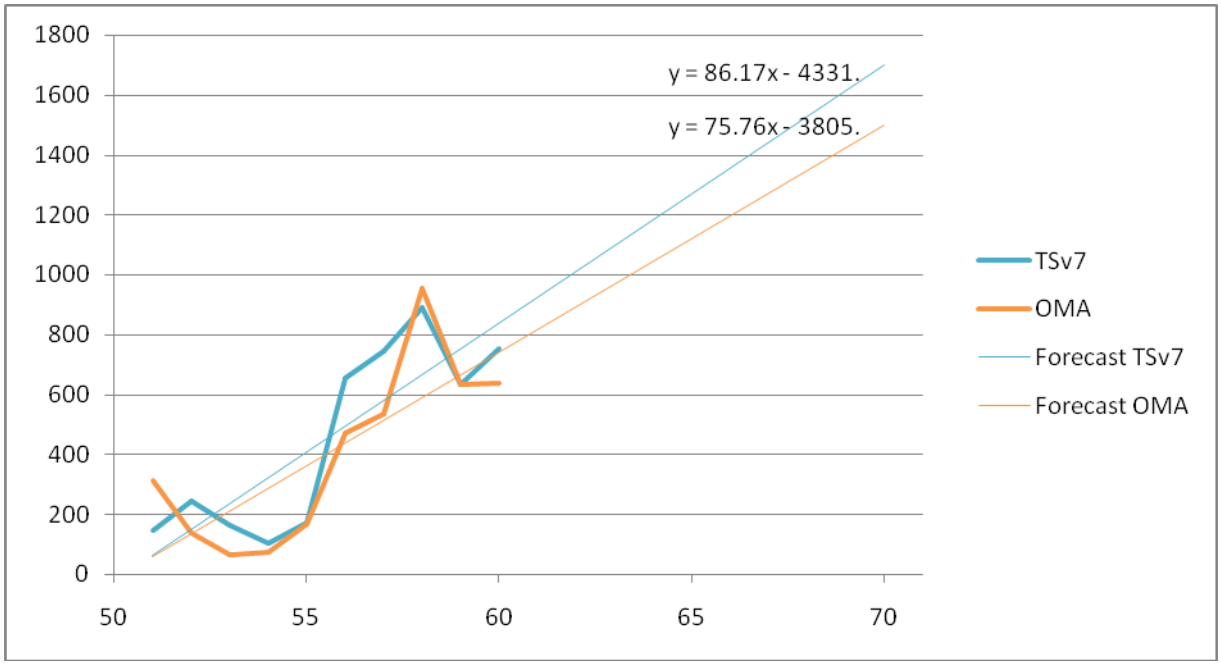

Figure 7. Average performance of TSv7, OMA and their forecasted values in seconds for each instance size L, normal scale.

We take higher forecast (achieved from TSv7 trend) as our time limit:

$$\mathrm{timeLimit(L)} = 86.172\mathrm{L} - 4331.4\ (\mathrm{s}). \tag{3}$$

Thus, time limit varies from 925 seconds for $\mathrm{L} = 61$, to 1700 seconds for $\mathrm{L} = 70$.

Table 4 and Figure 8 show results of this type of experiments.

Table 4. Average energy of best solution achieved by TSv7 and OMA for each instance size L.

L	TSv7	OMA
61	229.6	229.2
62	239	239.8
63	228.6	221
64	250.4	228
65	266.4	267.6

66	267.4	267.4
67	275	271.4
68	276	281.2
69	301.6	290.8
70	327	311

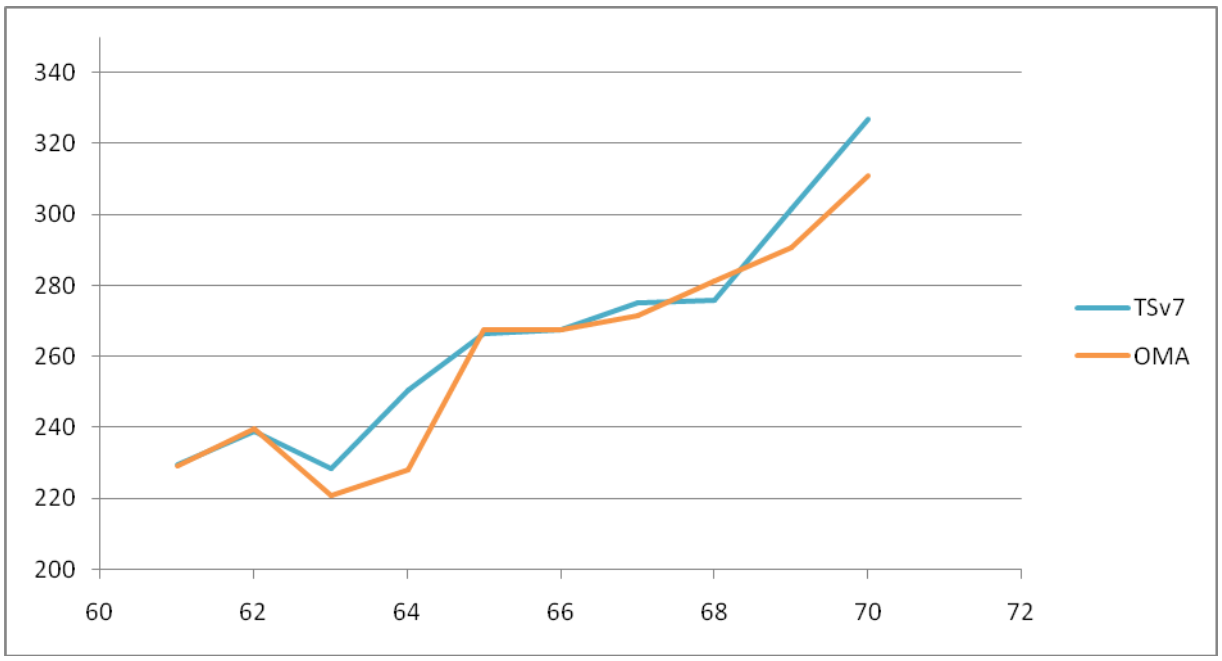

Figure 8. Average energy of best solution achieved by TSv7 and OMA for each instance size L.

As we can see performance of both approaches on larger instances also quite similar with some advantage of OMA in average. This shows that performance of both TSv7 and OMA changes identically with growth of complexity of problem. This is partially due to usage of TS local search mechanism as main component of bothmetaheuristics.

Conclusion

We showed importance of effective energy recomputation during stochastic local search applied to LABS problem.Progress in reducing time cost of these operator made by authors [6,7,8] allowed to bring LABS solvers to a new level of effectiveness. Though, metaheuristic strategy staid the same throughout these improvements – TS mechanism played key role in most of the approaches.

We developed new variant of memetic algorithm which incorporates best from both recent state-of-the-art approaches. As it performs on same level of effectiveness with latest TSv7 solver, we conclude that new LABS problem heuristics should use developments in energy recomputation to at least reach current state-of-the-art performance. Yet, comparison of different metaheuristic strategies applied to LABS problem is an open question.

Bibliography

[Gallardo, Cotta, Fernandez, 2009] J.E.Gallardo, C.Cotta, A.J.Fernandez. Finding Low Autocorrelation BinarySequences with Memetic Algorithms. In: Applied Soft Computing, 2009,vol. 9, pp. 1252–1256.

[Golay, 1982] M.J.E.Golay. The merit factor of long low autocorrelation binary sequences.In: IEEE Transactions on Information Theory, 1982,vol. 28(3), pp. 543–549.

[Mertens, 1996] S.Mertens. Exhaustive search for low-autocorrelation binary sequences.In: Journal of Physics A: Mathematical and General, 1996,vol. 29, pp. 473–481.

[Dotu, van Hentenryck, 2006] I.Dotu, P.vanHentenryck.A Note on Low Autocorrelation Binary Sequences. In:Constraint Programming, 2006, pp. 685–689.

[Gallardo, Cotta, Fernandez, 2007] J.E.Gallardo, C.Cotta, A.J.Fernandez.A Memetic Algorithm for the Low Autocorrelation Binary Sequence Problem. In: GECCO, 2007, pp. 1226–1233.

[Halim, Yap, Halim, 2008] S.Halim, R.Yap, F.Halim. EngineeringStochastic Local Search for the Low Autocorrelation Binary Sequence Problem. In: Principles and Practice of ConstraintProgramming, Australia, Sydney, 2008.

Authors' Information

Leonid Hulianytskyi– *Dr.Sc.(Technology), head of Department of combinatorial optimization and intelligent information technology of V.M.Glushkov Instituteof Cybernetics of National Academy of Sciences of Ukraine; Prof. of NTUU "KPI" and* Taras Shevchenko National University of Kyiv*; e-mail:*lh_dar@hotmail.com

Major Fields of Scientific Research: Combinatorial Optimization, Decision Making,Mathematical Modeling and practical applications

Vladyslav Sokol *– Master student in National Technical University of Ukraine "Kyiv Polytechnic Institute"; e-mail:* sokol.vladyslav@gmail.com

Major Fields of Scientific Research:General Computer Science, Combinatorial Optimization, Artificial Intelligence