A Memetic Algorithm for the Low Autocorrelation Binary Sequence Problem

José E. Gallardo
pepeg@lcc.uma.es

Carlos Cotta
ccottap@lcc.uma.es

Antonio J. Fernández
afdez@lcc.uma.es

Dept. de Lenguajes y Ciencias de la Computación.
ETSI Informática. University of Málaga.
Campus de Teatinos, 29071 - Málaga, Spain.

ABSTRACT

Finding binary sequences with low autocorrelation is a very hard problem with many practical applications. In this paper we analyze several metaheuristic approaches to tackle the construction of this kind of sequences. We focus on two different local search strategies, steepest descent local search (SDLS) and tabu search (TS), and their use both as stand-alone techniques and embedded within a memetic algorithm (MA). Plain evolutionary algorithms are shown to perform worse than stand-alone local search strategies. However, a MA endowed with TS turns out to be a state-of-the-art algorithm: it consistently finds optimal sequences in considerably less time than previous approaches reported in the literature.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Stochastic programming.

General Terms

Algorithms, Experimentation.

Keywords

Low autocorrelation binary sequences, memetic algorithms, tabu search.

1. INTRODUCTION

The *low autocorrelation binary sequence* (LABS) problem is a very hard combinatorial optimization problem. It has been deeply studied since the 1960s by both the communities of Physics and Artificial Intelligence. The reasons behind this interest are twofold: on one hand, the problem has many applications in diverse areas such as telecommunications (e.g., synchronization, pulse compression and, especially, radar), physics (e.g., ising spin glasses) and chemistry [1, 4, 17, 18, 19]; on the other hand, it poses a formidable optimization task of huge difficulty. In this sense, the application of many different techniques to the resolution of the problem has been attempted (see Sect. 2.2). For instance, analytical methods are able to find optimal sequences up to a certain length, but search methods are required in the most interesting cases (i.e., the aperiodic ones). In line with this, systematic search has been applied with limited success since, in general, these methods lack scalability, and therefore large sequences cannot be solved in a limited-resource (i.e., time and memory) scenario. To date, the largest LABS shown to be optimal is that with length 60, and this took several days of execution on a cluster of 160 CPUs [13].

In addition to complete deterministic approaches, stochastic methods have been also proposed to generate LABS, but in general they have performed poorly. One of the reasons is that the search landscape traditionally considered for finding LABS has a very irregular structure with isolated minima [1]. Contrarily to the general impression (as reported so far in the literature, see Sect. 2.2) that stochastic methods are not adequate to tackle this problem, this paper shows that these techniques are useful tools for finding optimal LABS. More precisely, we consider different local search strategies as well as memetic algorithms (MAs) [9, 10, 15] endowed with them. The combination of the global search performed by an evolutionary algorithm combined with the intensification provided by the local search component (in our case, a MA using tabu search) can successfully traverse the previously mentioned irregular landscape, and can consistently find optimal sequences in considerably less time than previous reported approaches.

The rest of the paper is organized as follows: In Section 2, the LABS problem is formally described and other approaches in the literature to tackle it are reviewed. In Section 3, two local search operators for the LABS problem are described. Section 4 presents the experimental evaluation of the local search operators described previously, along with MAs endowed with them. Finally, Section 5 concludes and outlines some future work.

2. BACKGROUND

In order to introduce the problem, let a binary sequence S of length L be represented by $s_1 s_2 \cdots s_L$ with $s_i \in \{-1, 1\}$ for $1 \leqslant i \leqslant L$, i.e., $S \in \{-1, 1\}^L$. Next subsection will define the notion of autocorrelation on such a sequence, as well as different quality measures. Subsequently, an overview of previous work on the LABS problem will be presented.

2.1 Low autocorrelation binary sequences

The *aperiodic autocorrelation* of elements in sequence S with distance k is defined as

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

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

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

and the low autocorrelation problem for binary sequences with length L, LABS(L), consists of finding a sequence of length L with associated minimum energy. In this sense, notice one interesting property of the LABS problem: it is highly symmetric. Clearly, the energy corresponding to a sequence remains unchanged when the sequence is reversed or complemented (i.e., every s_i is multiplied by -1). Note also that when alternate elements of a sequence get complemented, only the sign of odd-indexed correlations change, and hence, the corresponding energy is neither altered. Therefore, except for a small number of symmetric sequences, the 2^L sequences of length L come in equivalence classes of size 8.

Golay [7, 8] introduced a different measure in order to assess the quality of sequences called its *merit factor*:

$$F(S) = \frac{L^2}{2E(S)} \tag{3}$$

that lends itself to better analytical manipulation. If we define F_L to be the optimal value of the merit factor for sequences of length L, the LABS(L) problem can be alternatively defined as finding F_L such that:

$$F_L = \max_{S \in \{-1,1\}^L} F(S). \tag{4}$$

Based on an assumption termed *the ergodicity postulate*, Golay estimated an asymptotic value for F_L, namely $F_L \to 12.32$ for $L \to \infty$.

As a combinatorial problem, the search space for the LABS(L) problem has size 2^L, and the merit factor of a sequence can be computed in time $O(L^2)$. One of the hardness sources of the LABS problem is epistasis: different correlations $C_k(S)$ for a sequence S are not independent, and a change to one symbol s_i leading to an improvement of a certain $C_k(S)$ will affect the values of remaining correlations too. Another difficulty lies in the small number of global optima for most values of L, as it has been observed in cases for which solutions have been completely enumerated. The corresponding search landscape is dominated by a large number of local minima, while the global minima are extremely isolated deep and narrow holes [4, 6]. Presently, no analytical method exists for finding optimal sequences with minimal aperiodic correlations. To date, the only procedure to find the sequence with optimal F_L consists of using an implicit enumerative search among all 2^L possible sequences.

2.2 Related work

The LABS problem has been tackled in the literature using exact and heuristic methods. Golay [8] compiled the earlier works of Turyn [18] and Lindner [11], performing an exhaustive search enumeration to present optimal sequences for $L \leqslant 32$. Mertens used a parallel branch and bound with symmetry breaking procedures in [12] to solve instances up to $L = 48$. Using a four-processor Sun SPARCstation 20, it took a total 313 hours of CPU time to solve these instances. Recently, this algorithm has been used by Mertens and Bauke [13] to compute the optimal merit factors of sequences of length up to 60. Note that it took several days of execution on a cluster of 160 CPUs to solve the $L = 60$ instance. However, even with these enhancements, systematic search is unable to scale up to larger sequences.

For decades, approaches using stochastic methods on the LABS problem such as simulated annealing [1] and evolutionary search [3, 14] performed poorly with respect to finding optimal sequences. This is true even when restricting the search to a special kind of sequences, called *skew-symmetric* sequences, that allows the reduction of the search space by half (at the expense of not guaranteeing optimality for the general case). Recently, several stochastic algorithms have been reported to find optimal solutions. The first one was presented by Prestwich in [16], who was able to find global optima up to $L = 48$ with a hybrid algorithm (named CLS) that used local search and constraint programming. The algorithm was estimated to run in time $O(1.68^L)$, and the $L = 45$ instance was the one requiring more computing time to find the optimum (a mean time of 52,920 seconds on a 300MHz DEC Alphaserver 1000A 5/300). Dotú and Van Hentenryck presented in [5] a tabu search algorithm capable of solving $L \leqslant 48$ instances from 8 up to 55 times faster than CLS. In this case, the $L = 43$ instance was the one requiring more computing time to find the optimum (a mean time of 1,600 seconds on a 3.01 GHz PC). Finally, Brglez *et al.* [2] presented an evolutionary strategy (ES) and a Kernighan-Lin (KL) algorithm, that finds optimal values up to $L = 60$. With respect to computing time, KL performs better and is able to find the optimal solution in 68% of the runs for the $L = 48$ instance in 1,080 seconds (on a 266 MHz workstation). For $L = 60$, KL needed 20 hours for each run.

3. METAHEURISTICS FOR THE LABS PROBLEM

The LABS problem fits nicely with evolutionary algorithms (EAs), at least regarding off-the-shelf application. Since the problem does not pose constraints on the construction of solutions, sequences of length L can be naturally represented as binary strings in $\{-1, 1\}^L$, and blind operators for recombination and mutation can be readily used. Furthermore, there exists a well-defined objective function (to be minimized), i.e., the energy of a sequence as shown in equation (2). This said, such a less-principled approach cannot deal appropriately with the complexity of the problem, as it will be empirically shown in Sect. 4. For this reason, it is necessary to augment the EA with problem-aware add-ons. This can be accomplished via the use of embedded local search strategies, as described in the following.

3.1 Neighborhood Structure

In order to perform local search, we consider the neighborhood of a solution S with length L obtained by flipping exactly one symbol in the sequence, that can be expressed as

$$\mathcal{N}(S) = \{\mathit{flip}(S, i) \mid i \in \{1, .., L\}\} \tag{5}$$

where $\mathit{flip}(s_1 \cdots s_i \cdots s_L, i) = s_1 \cdots -s_i \cdots s_L$.

The evaluation of the fitness function is $O(L^2)$, and hence a naive implementation that completely recomputes the value of a solution after flipping a single symbol in a sequence S would be rather inefficient. A better implementation can be obtained by storing all computed products in a $(L-1)\times(L-1)$ table $\mathcal{T}(S)$, such that $\mathcal{T}(S)_{i\,j} = s_j s_{i+j}$ for $j \leqslant L-i$, and saving the values of the different correlations in a $L-1$ dimensional vector $\mathcal{C}(S)$, defined as $\mathcal{C}(S)_k = C_k(S)$ for $1 \leqslant k \leqslant L-1$. Fig. 1 shows these data structures for a $L=5$ instance.

$\mathcal{T}(S)$

	1	2	3	4
1	s_1s_2	s_2s_3	s_3s_4	s_4s_5
2	s_1s_3	s_2s_4	s_3s_5	
3	s_1s_4	s_2s_5		
4	s_1s_5			

$\mathcal{C}(S)$

1	$s_1s_2+s_2s_3+s_3s_4+s_4s_5$
2	$s_1s_3+s_2s_4+s_3s_5$
3	$s_1s_4+s_2s_5$
4	s_1s_5

Figure 1: Data structures to efficiently recompute fitness for a sequence of length 5, $S=(s_1,s_2,\cdots,s_5)$.

By observing that flipping a single symbol s_i multiplies by -1 the values of all cells in $T(S)$ where s_i is involved, the fitness of sequence *flip*(S,i) can be efficiently recomputed in time $O(L)$ as the result of the expression *ValueFlip*$(S,i,\mathcal{T}(\mathcal{S}),\mathcal{C}(\mathcal{S}))$, defined in Fig. 2.

```
function ValueFlip($S, i, \mathcal{T}, \mathcal{C}$)
  $f := 0$
  for $p := 1$ to $L-1$ do
    $v := \mathcal{C}_p$
    if $p \leqslant L-i$ then
      $v := v - 2\mathcal{T}_{p\,i}$
    end if
    if $p < i$ then
      $v := v - 2\mathcal{T}_{p\,(i-p)}$
    end if
    $f := f + v^2$
  end for
  return $f$
end function
```

Figure 2: Efficient recomputation of fitness for a move in local search.

3.2 Local Search strategies

Using the efficient fitness recomputation mechanism described before, two local search strategies have been defined. The first one we have considered is a *steepest descent local search* (SDLS) procedure, that moves to the best sequence in the neighborhood until reaching a local optimum. The pseudocode for this algorithm is depicted in Fig. 3

The second local search strategy considered uses tabu search (TS) as a mechanism to scape from local optima. For this purpose, we have used a L-dimensional vector $\mathcal{M}$

```
function SDLS($S, \mathcal{T}, \mathcal{C}$)
  $S^* := S$
  $f^* := \sum_{k=1}^{L-1} \mathcal{C}_k^2$
  repeat
    $f^\dagger := \infty$
    for $i := 1$ to $L$ do /* search best move */
      $S' := flip(S^*, i)$
      $f' := ValueFlip(S^*, i, \mathcal{T}, \mathcal{C})$
      if $(f' < f^\dagger)$ then
        $f^\dagger := f'$
        $S^\dagger := S'$
      end if
    end for
    if $(f^\dagger < f^*)$ then
      $S^* := S^\dagger$
      $f^* := f^\dagger$
      improvement := True
      update $\mathcal{T}$ and $\mathcal{C}$
    else
      improvement := False
    end if
  until not improvement
  return $S^*$
end function
```

Figure 3: Steepest descent local search (SDLS) procedure for the LABS problem.

as an attributive recency-based memory, so that if $\mathcal{M}_i = k$, flipping the i-th symbol in the current sequence is forbidden until the k-th iteration of the search. The *aspiration criteria* for ignoring tabu moves is improving the best solution found in the current run of the local search. The actual pseudocode of this procedure is shown in Fig. 4. For each iteration, the search moves to the best sequence in the current neighborhood that is not tabu, and the corresponding flipped attribute is forbidden for a random number of iterations proportional to the value of *maxIters*.

4. EXPERIMENTAL RESULTS

All algorithms have been run for different instance sizes $L \in [3,60]$, corresponding to the instances for which optimal solutions are known (see Table 1 for optimal merit factors for the larger instances). Twenty independent executions have been performed for each algorithm and instance size. The termination criteria for each execution has been either finding the optimal solution or reaching a time limit. This limit has been set to 5 minutes for $L \leqslant 30$, and has been gradually incremented in one minute for each size increment for $L > 30$ (i.e. the greatest time limit was 35 minutes for $L = 60$). All experiments have been performed on a 2.4GHz P4 PC under Linux.

First of all, experiments have been carried out with a steady state EA (*popsize* $= 100, p_m = 1/L, p_X = 0.9$, binary tournament selection, uniform crossover) that did not perform local search. A full description of the distribution of results (as the relative distance to the optimum) is shown in Fig. 5. For $L < 25$, the algorithm has been able to find the optimal solution in all runs, but note how the performance degrades when L is increased (for $L > 38$ the EA

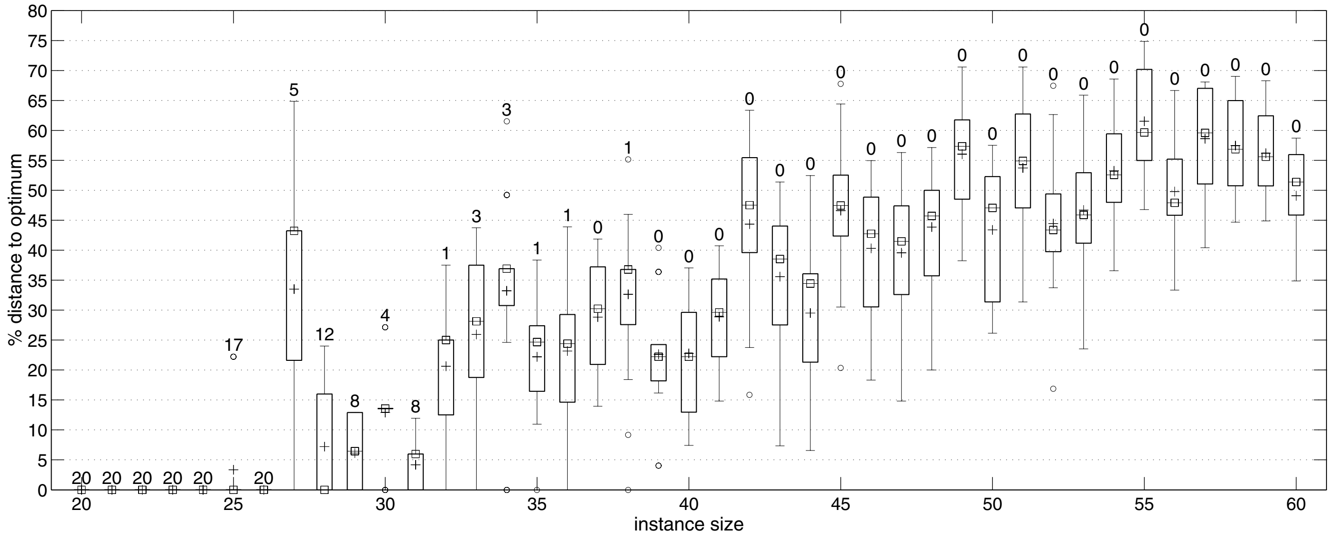


Figure 5: Relative distance to optimum for the EA and different instance sizes. In all box plots, the figure above indicates the number of runs (out of 20) leading to an optimal solution. A + sign indicates the mean of the distribution, whereas a □ marks its median. The bottom and top of the box correspond respectively to the first and third quartile of the distribution. *Outliers* are indicated by small circles in the plot.

```
     function TabuSearch($S, \mathcal{T}, \mathcal{C}$)
 1:    $\mathcal{M}_i := 0$, for $1 \leqslant i \leqslant L$ /* initialize tenure table */
 2:    $minTabu := maxIters/10$
 3:    $extraTabu := maxIters/50$
 4:    $S^* := S$
 5:    $f^* := \sum_{k=1}^{L-1} \mathcal{C}_k^2$
 6:    for $k := 1$ to $maxIters$ do
 7:      $f^\dagger := \infty$
 8:      for $i := 1$ to $L$ do /* search best move */
 9:        $S' := flip(S, i)$
10:        $f' := ValueFlip(S, i, \mathcal{T}, \mathcal{C})$
11:        if $(k \geqslant \mathcal{M}_i)$ or $(f' < f^*)$ then
12:          if $(f' < f^\dagger)$ then
13:            $f^\dagger := f'$
14:            $S^\dagger := S'$
15:            $i^\dagger := i$
16:          end if
17:        end if
18:      end for
19:      $S := S^\dagger$ /* make move */
20:      update $\mathcal{T}$ and $\mathcal{C}$
21:      $\mathcal{M}_{i^\dagger} := k + minTabu + URand[0, extraTabu)$
22:      if $(f^\dagger < f^*)$ then
23:        $S^* := S^\dagger$
24:        $f^* := f^\dagger$
25:      end if
26:    end for
27:    return $S^*$
     end function
```

Figure 4: Tabu search procedure for the LABS problem.

has been unable to find a single optimal solution, and the relative distance to the optimum is large).

Table 1: Energy and merit factor for optimal solutions and different instance sizes

Instance size	Energy	Merit factor	Instance size	Energy	Merit factor
39	99	7.68	50	153	8.16
40	108	7.40	51	153	8.50
41	108	7.78	52	166	8.14
42	101	8.73	53	170	8.26
43	109	8.48	54	175	8.33
44	122	7.93	55	171	8.84
45	118	8.58	56	192	8.16
46	131	8.07	57	188	8.64
47	135	8.18	58	197	8.53
48	140	8.22	59	205	8.49
49	136	8.82	60	218	8.25

Next, experiments have been done to measure the performance of local search procedures. Both TS and SDLS have been embedded into a random restart driving procedure, that beginning from a random configuration performed independent repetitions of the local search procedures[1]. Algorithms are restarted until the time limit is reached, and the best solution found is returned. These experiments aim to set the baseline for further comparison to MAs endowed with both procedures. Fig. 6 and Fig. 7 show the result of these experiments (distributions for $L < 40$ are omitted as both algorithms were able to find optimal solutions in all

[1]SDLS is run in each case until locating a local optima. In the case of TS, we have used a value of *maxIters* drawn from $[L/2, 3L/2]$. We also tested longer runs, coupled with intensification strategies that returned to the incumbent of the run, but the results did not improve those of the random restarting strategy. We hypothesize that this is due to the rugged structure of the fitness landscape, that benefits in this particular case restarting over intensification.

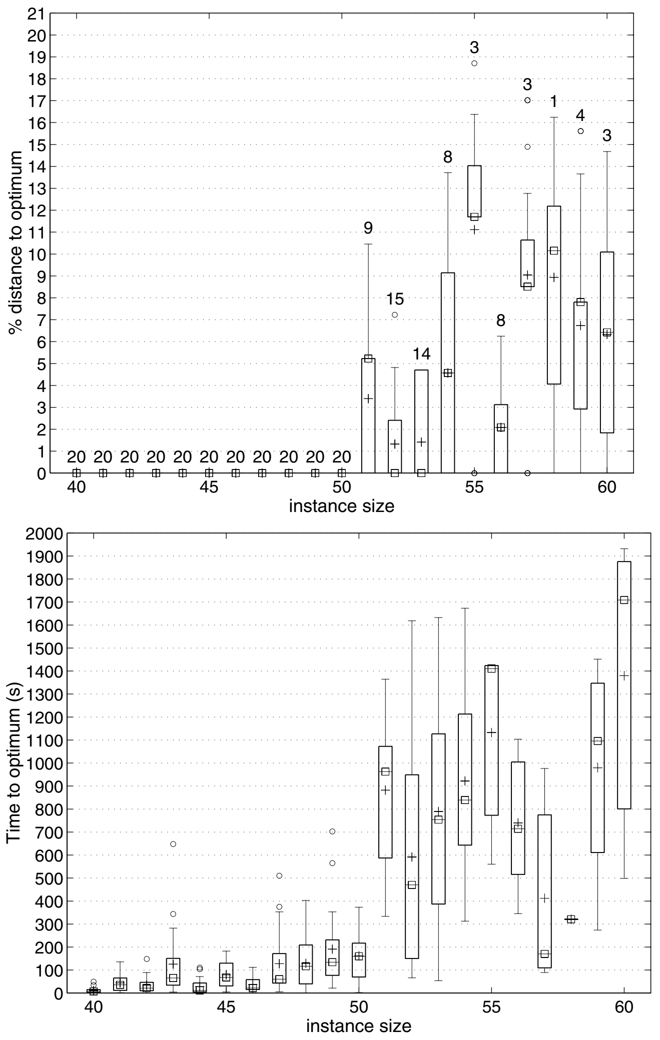


Figure 6: Relative distance to the optimum (top) and time to find the optimum in seconds (bottom) for SDLS and different instance sizes.

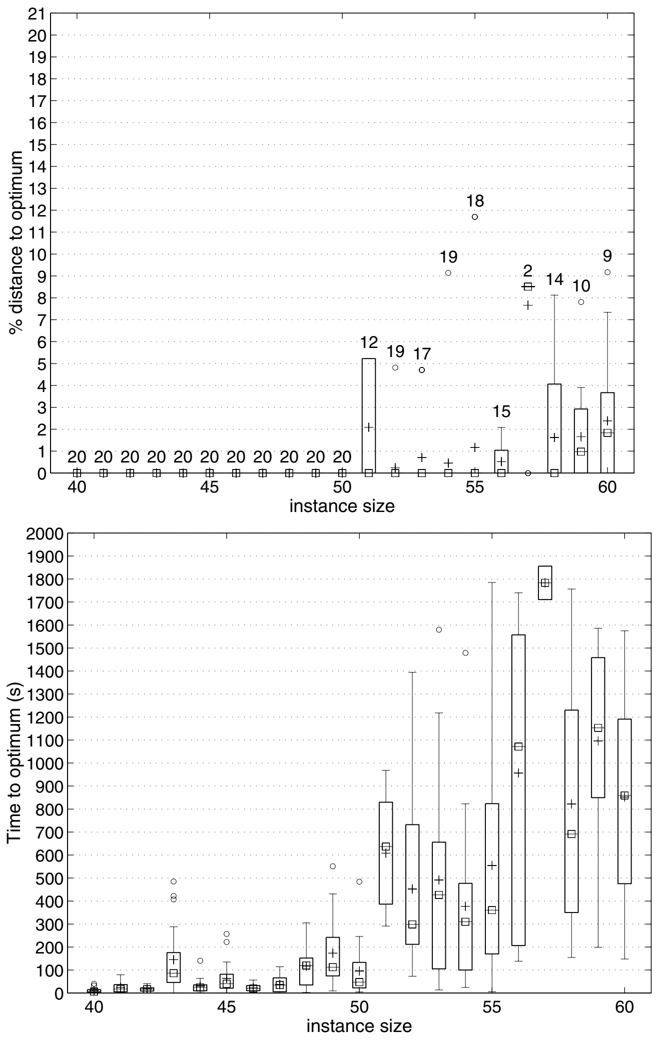


Figure 7: Relative distance to the optimum (top) and time to find the optimum in seconds (bottom) for TS and different instance sizes.

```
for $i := 1$ to $popsize$ do
  $pop[i]$ := RANDOM BINARY SEQUENCE($L$)
  EVALUATE($pop[i]$)
end for
while allowed runtime not exceeded do
  for $i := 1$ to $offsize$ do
    if recombination is performed then
      $parent_1$ := SELECT($pop$)
      $parent_2$ := SELECT($pop$)
      $offspring[i]$ := RECOMBINE($parent_1$, $parent_2$)
    else
      $offspring[i]$ := SELECT($pop$)
    end if
    if mutation is performed then
      $offspring[i]$ := MUTATE($offspring[i]$)
    end if
    $offspring[i]$ := LOCAL SEARCH($offspring[i]$)
    Evaluate($offspring[i]$)
  end for
  $pop$ := REPLACE($pop$, $offspring$)
end while
```

Figure 8: Pseudocode of the memetic algorithm.

Table 2: Comparison of MA$_{\text{TS}}$ and tabu search algorithm in [5]. Table shows mean time in seconds to find the optimum for both algorithms.

instance size	TS [5] time	% success	MA$_{\text{TS}}$ time	% success	(mean time) speedup
40	260.11	100	3.67	100	70.97
41	460.26	100	19.79	100	23.26
42	466.73	100	9.76	100	47.82
43	1600.63	100	51.56	100	31.04
44	764.66	100	21.56	100	35.47
45	1103.48	100	24.77	100	44.55
46	703.32	100	8.34	100	84.37
47	1005.03	100	13.27	100	75.72
48	964.13	100	56.86	100	16.96

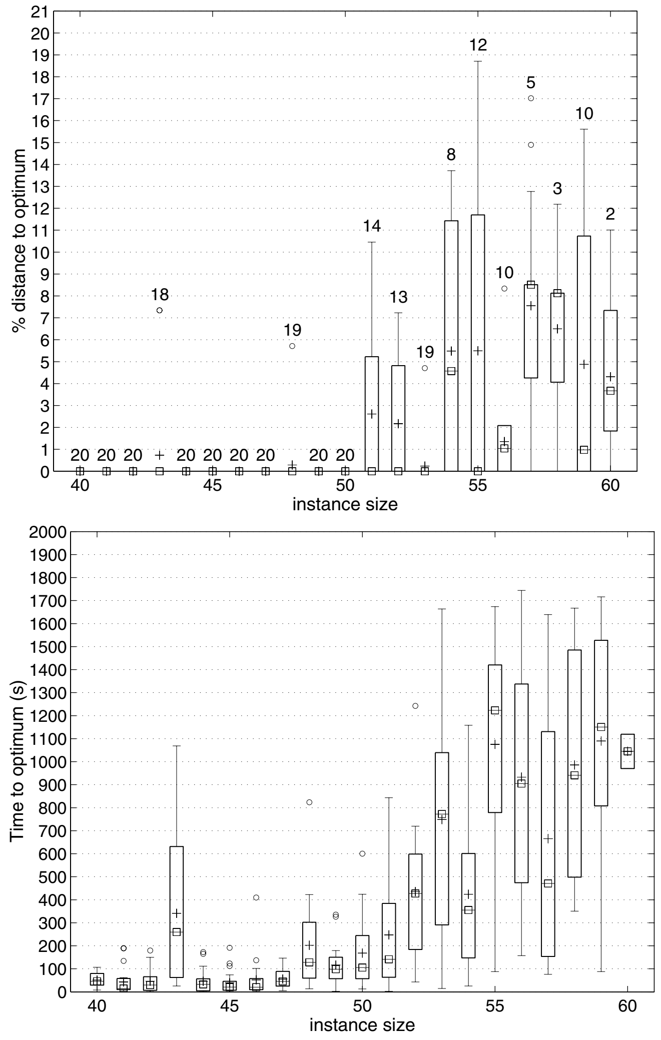
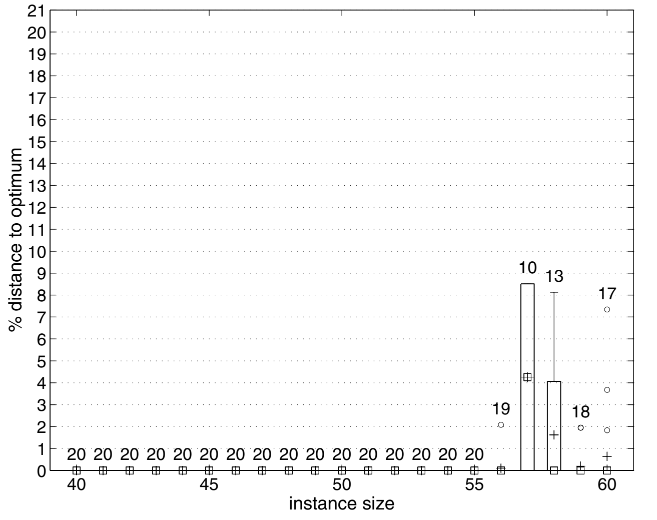


Figure 9: Relative distance to the optimum (top) and time to find the optimum in seconds (bottom) for MA$_{\text{SDLS}}$ and different instance sizes.

runs in a few seconds). Observe that TS and SDLS can find optimal solutions consistently for $L \leqslant 50$. For $L > 50$, although SDLS is able to find the optimum in at least one run for all instance sizes, it is not robust in most cases. The performance of TS is clearly better, finding optimal solutions in at least 50% of the runs for all instances except for $L \in \{57, 60\}$.

In subsequent experiments, the performance of two memetic algorithms (see Fig. 8 for a pseudocode) endowed with SDLS and TS (denoted MA$_{\text{SDLS}}$ and MA$_{\text{TS}}$ respectively) has been empirically analyzed. The underlaying algorithm is the same as the EA described above. Results are shown in Fig. 9 and Fig. 10 (again for $L \geqslant 40$, since the remaining instances are easily solved). Although MA$_{\text{SDLS}}$ performs better than SDLS alone in most cases (showing the benefit of embedding the local search operator within a MA), it still performs weakly for large instances.

In general, the best overall results are obtained by MA$_{\text{TS}}$. This algorithm showed a high robustness and significantly improved the execution times of the best approaches reported in the literature that tackled the LABS problem. Regarding execution times, notice that these have been reported differently for the TS algorithm presented in [5] and the KL algorithm described in [2]. In [5], Dotú and Van Hentenryck provided the mean time to find the optimum with respect all the runs, whereas Brglez *et al.* reported in [2] the allowed execution time for each run. Due to this difference we compared the execution times of MA$_{\text{TS}}$ with those informed in [5] and [2] separately. Compared to the TS algorithm presented in [5], Table 2 shows that, like the TS algorithm, MA$_{\text{TS}}$ also finds optimal solutions in all the runs for $L \leqslant 48$ but this is done in a mean time lower than 57 seconds in the worse case (i.e., $L = 48$); in fact, MA$_{\text{TS}}$ is between 5 (for some instances $L < 40$) and 84 times faster (running on a 20% slower machine) than Dotú and Van Hentenryck's TS. Compared to [2], the KL algorithm finds the optimum for the $L = 60$ instance in 20 hours of execution time, whereas MA$_{\text{TS}}$ needs a mean time of 875 seconds, which implies a speed up of about one order of magnitude when the computation power of the different platforms are



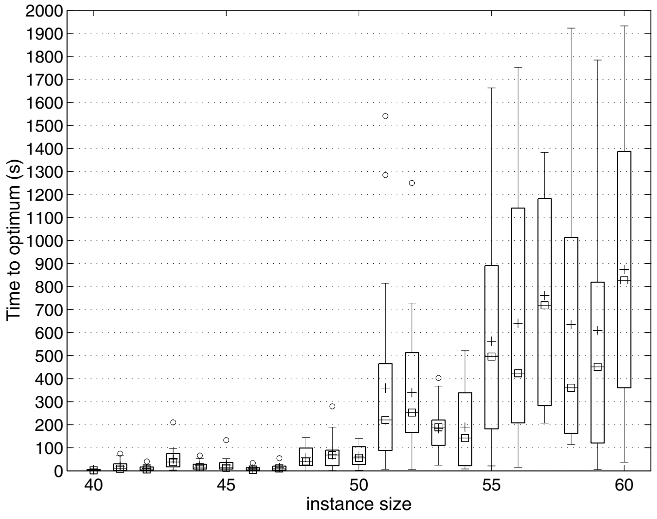


Figure 10: Relative distance to the optimum (top) and time to find the optimum in seconds (bottom) for MA$_{\text{TS}}$ and different instance sizes.

adjusted. Moreover, Table 3 shows the allowed execution time (in seconds) for both KL and MA$_{\text{TS}}$ algorithms when $L \geq 48$: observe again that the speedup of MA$_{\text{TS}}$ with respect to KL is more than one order of magnitude in the larger instances ($57 \leq L \leq 60$).

Also, as shown in Tables 2 and 3, the MA$_{\text{TS}}$ algorithm is very robust, finding optimal solutions in all runs for $L \leqslant 48$ and clearly outperforming KL in the instances $48 \leq L \leq 55$, (obtaining again the optimum in all the runs). In addition, for the larger instances ($L \geq 56$), MA$_{\text{TS}}$ achieves optimal solutions in most executions (i.e., topping 85%), except for $L \in \{57, 58\}$, for which the success ratios are 50% and 65% (in these cases, the mean distances to the optimum are 4.25% and 1.6%). Observe however that in these higher instances the relation *success/time* is clearly favorable to MA$_{\text{TS}}$. For example, assuming a number of independent runs of MA$_{\text{TS}}$ summing up the same computational time (adjusted for platform differences) than one run of KL, we can compute the equivalent success ratio of MA$_{\text{TS}}$ for size 57 and 58 as 85% and 87% respectively. This way, MA$_{\text{TS}}$ can be shown to perform at the state-of-the-art level for the LABS problem.

Table 3: Comparison of MA$_{\text{TS}}$ and KL algorithm in [2]. Table shows allowed execution time in seconds and percentage of success for both algorithms.

instance size	KL [2] time	% success	MA$_{\text{TS}}$ time	% success	(allowed time) speedup
48	1080	68	1380	100	0.78
49	1440	75	1440	100	1.00
50	2160	93	1500	100	1.44
51	2880	31	1560	100	1.85
52	4320	75	1620	100	2.67
53	6120	75	1680	100	3.64
54	8640	62	1740	100	4.96
55	12600	87	1800	100	7.00
56	18000	100	1860	95	9.67
57	47520	68	1920	50	24.75
58	35280	81	1980	65	17.81
59	50040	100	2040	90	24.52
60	72000	100	2100	85	34.28

5. CONCLUSIONS AND FUTURE WORK

We have shown that evolutionary methods can successfully compete with (and often outperform) most approaches existing to date for the LABS problem. Particularly, we have provided empirical evidence that –despite EAs can be straightforwardly deployed on the LABS problem– pure evolutionary approaches cannot cope with the complexity of the problem. They require the assistance of local-search operators to provide optimal or near-optimal results consistently. To this end, we have considered two local search strategies, namely steepest descent local search and tabu search. The results indicate that embedding them within the EA improves synergistically the search capabilities of the algorithm. Furthermore, the computational time required for finding optimal solutions in previous state-of-the-art heuristic approaches is improved by one order of magnitude.

As to future extensions, work is underway in several lines. Firstly, we plan to apply the MA$_{\text{TS}}$ algorithm to larger instances for which the optimal is not known, in order to test the scalability of the approach in the long term (preliminary results with $L \leqslant 70$ indicate that the algorithm is capable of systematically recovering best-known solutions; experiments will be completed with larger values of L). Secondly, and related to the previous issue, we intend to adapt the algorithm to explore only skew-symmetric solutions [8]. This would allow testing the algorithm on even larger instances without excessively incrementing computing time.

6. ACKNOWLEDGMENTS

This work was partially supported by Spanish MCyT under contracts TIN2004-7943-C04-01 and TIN2005-08818-C04-01.

7. REFERENCES

[1] J. Bernasconi. Low autocorrelation binary sequences: statistical mechanics and configuration state analysis. *J. Physique*, 48:559–567, 1987.

[2] F. Brglez, X. Y. Li, M. F. Stallman, and B. Militzer. Reliable cost prediction for finding optimal solutions to labs problem: Evolutionary and alternative algorithms. In *Fifth International Workshop on Frontiers in Evolutionary Algorithms*, Cary, NC, USA, 2003.

[3] C. de Groot, D.Würtz, and K. Hoffmann. Low autocorrelation binary sequences: exact enumeration and optimization by evolutionary strategies. *Optimization*, 23:369–384, 1992.

[4] V. de Oliveira, J. Fontanari, and P. Stadler. Metastable states in high order short-range spin glasses. *J. Phys. A: Math. Gen.*, 32:8793–8802, 1999.

[5] I. Dotú and P. V. Hentenryck. A note on low autocorrelation binary sequences. In F. Benhamou, editor, *12th International Conference on Principles and Practice of Constraint Programming (CP 2006)*, volume 4204 of *Lecture Notes in Computer Science*, pages 685–689, Nantes, France, September 2006. Springer.

[6] F. Ferreira, J. Fontanari, and P. Stadler. Landscape statistics of the low autocorrelated binary string problem. *J. Phys. A: Math. Gen.*, 33:8635–8647, 2000.

[7] M. Golay. Sieves for low autocorrelation binary sequences. *IEEE Transactions on Information Theory*, 23:43–51, 1977.

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

[9] W. E. Hart, N. Krasnogor, and J. E. Smith (eds.). *Recent Advances in Memetic Algorithms*, volume 166 of *Studies in Fuzziness and Soft Computing*. Springer, 2004.

[10] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.

[11] J. Lindner. Binary sequences up to length 40 with best possible autocorrelation function. *Electron. Lett.*, 11:507, 1975.

[12] S. Mertens. Exhaustive search for low-autocorrelation binary sequences. *J. Phys. A: Math. Gen.*, 29::473–481, 1996.

[13] S. Mertens and H. Bauke. Ground states of the bernasconi model with open boundary conditions. Available on line in *http://odysseus.nat.uni-magdeburg.de/∼mertens/bernasconi/open.dat*, November 2004.

[14] B. Militzer, M. Zamparelli, and D. Beule. Evolutionary search for low autocorrelated binary sequences. *IEEE Transactions on Evolutionary Computation*, 2(1):34–39, 1998.

[15] P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 219–234. McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999.

[16] S. Prestwich. A hybrid local search for low autocorrelation binary sequences. Technical Report TR0001, Department of Computer Science, National University of Ireland, Cork, Ireland, 2000.

[17] P. Stadler. Landscapes and their correlation functions. *J. Math. Chem.*, 20(1-45), 1996.

[18] R. Turyn. Sequences with small correlation. In H. Mann, editor, *Error Correcting Codes*, pages 195–228. Wiley, New York, 1968.

[19] R. Turyn and J. Storer. On binary sequences. *Proc. Amer. Math. Soc.*, 12:394–399, 1961.