Finding Low Autocorrelation Binary Sequences with Memetic Algorithms

José E. Gallardo, Carlos Cotta, and Antonio J. Fernández

Department Lenguajes y Ciencias de la Computación*, Málaga University, Campus de Teatinos, 29071 - Málaga, Spain.* {*pepeg,ccottap,afdez*}*@lcc.uma.es.*

Abstract

This paper deals with the construction of binary sequences with low autocorrelation, a very hard problem with many practical applications. The paper analyzes several metaheuristic approaches to tackle this kind of sequences. More specifically, the paper provides an analysis of different local search strategies, used as standalone techniques and embedded within memetic algorithms. One of our proposals, namely a memetic algorithm endowed with a Tabu Search local searcher, performs at the state-of-the-art, as it consistently finds optimal sequences in considerably less time than previous approaches reported in the literature. Moreover, this algorithm is also able to provide new best-known solutions for large instances of the problem. In addition, a variant of this algorithm that explores only a promising subset of the whole search space (known as skew-symmetric sequences) is also analyzed. Experimental results show that this new algorithm provides new best-known solutions for very large instances of the problem.

Key words: Low autocorrelation binary sequences, memetic algorithms, tabu search, combinatorial optimization.

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 [1–5]; on the other hand, it poses a formidable optimization task of huge difficulty. In this sense, the application of many different techniques

(ranging from complete deterministic approaches to stochastic methods) to the resolution of the problem has been attempted (see Sect. 2.2).

Regarding stochastic methods, the general impression is that these techniques are not adequate to tackle this problem. In fact, considering the results reported in the literature this might be considered as a reality. In [6], we provided empirical evidence that this is the case for pure evolutionary algorithms (EAs), as they cannot cope with the complexity of the problem. Nonetheless, we also shown that evolutionary methods assisted by local-search operators (i.e., memetic algorithms [7–10], MAs) provide optimal or near-optimal results consistently for LABS instances whose optimal solutions are known to date. To this end, a steepest descent local search procedure and another one based on Tabu Search [11] (TS) were analyzed. Experimental results indicated that embedding them within a steady state EA improves synergistically the search capabilities of the algorithm and that these MAs (specifically the one embedding Tabu Search) can successfully compete with, and often outperform, current approaches for the LABS problem, as they provide optimal results in considerable less time.

We extend here our previous research on this problem. As new contributions, we firstly present a more extensive experimental analysis of the previously mentioned MAs on up-to-date solved instances, doubling the number of runs for each algorithm –thus enhancing the significance of results–, testing new recombination operators, and describing in more detail the observed behavior of the algorithms utilized. Secondly, in order to test the scalability of our approach, the best performing MA has been run on large instances of the LABS problem for which optimal solutions are currently unknown. Results were very successful, as this MA was able to provide new best-known solutions for many of these instances. Finally, we present an adaptation of this MA that only explores a promising sieve of the whole search space. To be precise, we considered so-called *skew-symmetric sequences* [12], that reduces the size of the search space explored by the algorithm at the expense of not guaranteing optimality for the general case of the problem. Experiments for this latter heuristic on very large instances show that it performs at the state-of-the-art, as it provides solutions better than or equal to those reported in the literature.

The rest of this paper is structured as follows: In Sect. 2, the LABS problem is formalized and previous work on it is surveyed. In Sect. 3, a neighborhood structure for the LABS problems is described, along with two local search algorithms to explore it. In Sect. 4, an exhaustive experimental evaluation of those local search algorithms and of memetic algorithms endowed with them is presented. Results for these algorithms are compared to current state of the art approaches for the LABS problem. In Sect. 5, one of the proposed memetic algorithms is adapted to search only in the space of so called *skew symmetric* sequences, a promising sieve of the whole search space for the LABS

problem. This algorithm is experimentally evaluated on very large instances of the problem. Finally, Sect. 6 presents conclusions for this work.

2 Background

In this section, we will define the notion of autocorrelation on binary sequences, as well as different measures that can be used to evaluate their quality. Subsequently, an overview of previous work on the LABS problem will be presented.

2.1 Low Autocorrelation Binary Sequences

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., $S \in \{-1, 1\}^L$. 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., when 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 [12, 13] 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 [5], and although it has been conjectured that global minima are extremely isolated deep and narrow holes [3, 14], no evidence for this was found in the study performed in [15]. 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. Systematic search has been applied with limited success. For instance, Golay published for the first time in [12] the optimal solutions for $L \leqslant 32$ that Lindner had computed by performing an exhaustive search enumeration. Mertens used a parallel branch and bound with symmetry breaking procedures in [16] 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 [17] 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. In general, these methods lack scalability, and therefore large sequences cannot be solved in a limited-resource (i.e., time and memory) scenario.

In addition to complete deterministic approaches, stochastic methods have been also proposed to generate LABS, but in general they have performed poorly. For decades, approaches using stochastic methods on the LABS problem such as simulated annealing [3] and evolutionary search [18–20] performed

poorly with respect to finding optimal sequences. Recently, several stochastic algorithms have been reported to find optimal solutions though. The first one was presented by Prestwich in [21], 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 [22] 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.* [23] 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.

For very large instances, best results are obtained by Militzer *et. al* in [14] using a (μ, λ)-ES algorithm. One important characteristic of this evolutionary algorithm is that no recombination is performed to generate the offspring. As to the mutation operator, it is not blind and exploits problem knowledge through a so-called *preselection* procedure, an heuristic that tries to diminish aperiodic autocorrelations (C_k) with large values. For this purpose, a subset of t autocorrelations is first randomly selected, and from this subset, the largest m autocorrelations are taken. Afterwards, $n > 1$ bits are flipped randomly, and the new solution is accepted if all m autocorrelations in the latter subset have been reduced. Otherwise, this procedure is iterated until a solution is accepted or a maximum number of trials ($s_{max} \approx L$) is reached.

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.

	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			

$\mathcal{T}(S)$

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

$\mathcal{C}(S)$

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

3.1 Neighborhood Structure

```
    function ValueFlip(S, i, T, C)
1:      f := 0
2:      for p := 1 to L − 1 do
3:          v := C_p
4:          if p ⩽ L − i then v := v − 2T_{p i} end if
5:          if p < i then v := v − 2T_{p (i−p)} end if
6:          f := f + v^2
7:      end for
8:      return f
    end function
```

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

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. This neighborhood can be expressed constructively 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.

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 $\mathit{flip}(S, i)$ can be efficiently recomputed in time $O(L)$ as the result of the expression $\mathit{ValueFlip}(S, i, \mathcal{T}(\mathcal{S}), \mathcal{C}(\mathcal{S}))$, defined in Fig. 2.

```
    function SDLS(S, T, C)
 1:     S* := S; f* := Σ_{k=1}^{L-1} C_k^2
 2:     repeat
 3:       f† := ∞
 4:       for i := 1 to L do /* search best move */
 5:         S' := flip(S*, i)
 6:         f' := ValueFlip(S*, i, T, C)
 7:         if (f' < f†) then f† := f'; S† := S' end if
 8:       end for
 9:       if (f† < f*) then
10:         S* := S†; f* := f†; improvement := True
11:         update T and C
12:       else
13:         improvement := False
14:       end if
15:     until not improvement
16:     return S*
    end function
```

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

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 as a mechanism to scape from local optima. For this purpose, we have used a L-dimensional vector $\mathcal{M}$ 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*.

```
    function TabuSearch(S, T, C)
1:      M_i := 0, for 1 ⩽ i ⩽ L /* initialize tenure table */
2:      minTabu := maxIters/10
3:      extraTabu := maxIters/50
4:      S* := S; f* := Σ_{k=1}^{L-1} C_k^2
5:      for k := 1 to maxIters do
6:        f† := ∞
7:        for i := 1 to L do /* search best move */
8:          S' := flip(S, i)
9:          f' := ValueFlip(S, i, T, C)
10:         if (k ⩾ M_i) or (f' < f*) then
11:           if (f' < f†) then
12:             f† := f'; S† := S'; i† := i
13:           end if
14:         end if
15:       end for
16:       S := S† /* make move */
17:       update T and C
18:       M_{i†} := k + minTabu + URand[0, extraTabu)
19:       if (f† < f*) then S* := S†; f* := f† end if
20:     end for
21:     return S*
    end function
```

Fig. 4. Tabu Search procedure for the LABS problem.

4 Experimental Results

All algorithms have been run for different instance sizes. Forty 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 2.4 GHz P4 PC under Linux.

4.1 Instances with Known Optima

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 (the pseudo code for this EA corresponds to the MA presented in Fig. 5, but without performing local search).

```
1:  for i := 1 to popsize do
2:    pop[i] := Random Binary Sequence(L)
3:    Evaluate(pop[i])
4:  end for
5:  while allowed runtime not exceeded do
6:    for i := 1 to offsize do
7:      if recombination is performed (p_X) then
8:        parent_1 := Select(pop)
9:        parent_2 := Select(pop)
10:       offspring[i] := Recombine(parent_1, parent_2)
11:     else
12:       offspring[i] := Select(pop)
13:     end if
14:     if mutation is performed (p_m) then
15:       offspring[i] := Mutate(offspring[i])
16:     end if
17:     offspring[i] := Local Search(offspring[i])
18:     Evaluate(offspring[i])
19:   end for
20:   pop := Replace(pop, offspring)
21: end while
```

Fig. 5. Pseudo code of the memetic algorithm.

A full description of the distribution of results (as the relative distance to the optimum) is shown in Fig. 6. For $L < 25$, the algorithm has been able to find the optimal solution in almost all runs, but note how the performance degrades when L is increased (for $L > 38$ the EA was only able to find two optimal solutions and the relative distance to the optimum is large).

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. 7 and Fig. 8, and Table 1 show the results of these experiments (distributions for $L < 40$ are omitted as both algorithms were able to find optimal solutions in all runs in a few seconds). Observe that TS

[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.

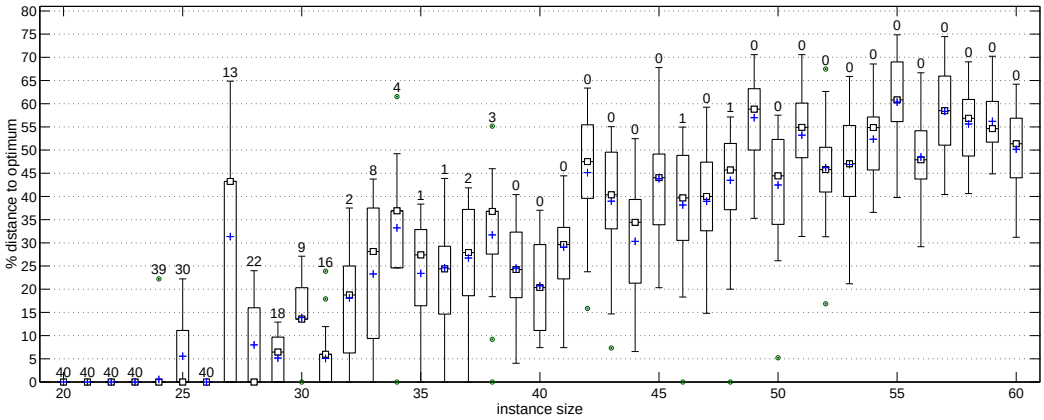


Fig. 6. Relative distance to optimum for the EA and different instance sizes. In all box plots, the figure above indicates the number of runs out of 40 leading to an optimal solution. A + sign indicates the mean of the distribution, whereas a □ marks its median. Boxes comprise the second and third quartiles of the distribution. *Outliers* are indicated by small circles in the plot.

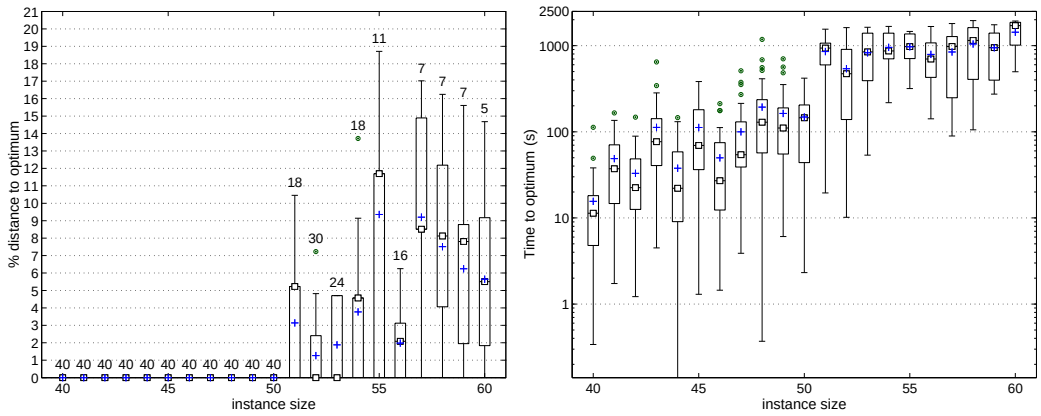


Fig. 7. Relative distance to the optimum (left) and time to find the optimum in seconds (right) for SDLS and different instance sizes.

and SDLS can find optimal solutions consistently for $L \leqslant 50$. For $L > 50$, although SDLS is able to find the optimum in several runs 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 $L = 57$.

In subsequent experiments, the performance of two memetic algorithms endowed with SDLS and TS (denoted MA_{SDLS} and MA_{TS} respectively) has been empirically analyzed. The underlaying algorithm is the same as the EA described above but with an additional local search phase, and its pseudo code is displayed in Fig. 5. Results are shown in Fig. 9 and Fig. 10, and in Table 2 (again for $L \geqslant 40$, since the remaining instances are easily solved). Although MA_{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

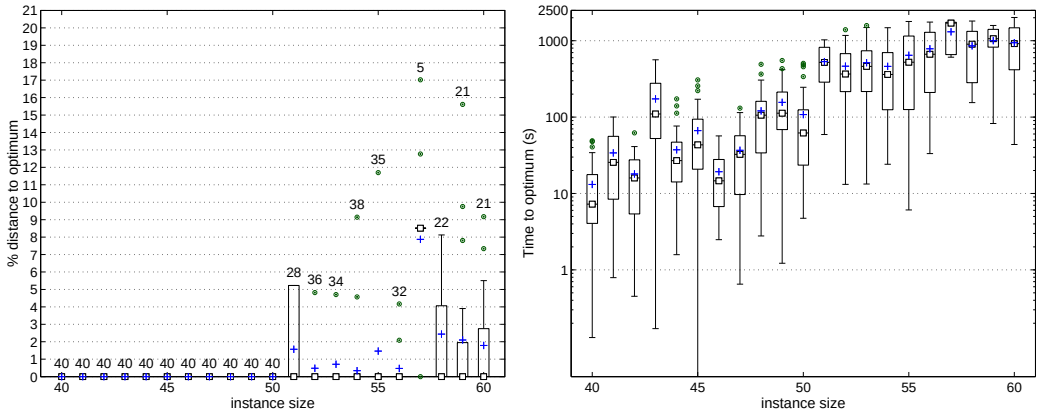


Fig. 8. Relative distance to the optimum (left) and time to find the optimum in seconds (right) for TS and different instance sizes.

Table 1
Numerical distribution of merit factors for SDLS and TS on different instance sizes. Optimal solutions are shown in boldface.

	SDLS			TS		
Instance	Best	Mean $\pm$ σ	Median	Best	Mean $\pm$ σ	Median
40	**7.40**	**7.40** $\pm$ 0.00	**7.40**	**7.40**	**7.40** $\pm$ 0.00	**7.40**
41	**7.78**	**7.78** $\pm$ 0.00	**7.78**	**7.78**	**7.78** $\pm$ 0.00	**7.78**
42	**8.73**	**8.73** $\pm$ 0.00	**8.73**	**8.73**	**8.73** $\pm$ 0.00	**8.73**
43	**8.48**	**8.48** $\pm$ 0.00	**8.48**	**8.48**	**8.48** $\pm$ 0.00	**8.48**
44	**7.93**	**7.93** $\pm$ 0.00	**7.93**	**7.93**	**7.93** $\pm$ 0.00	**7.93**
45	**8.58**	**8.58** $\pm$ 0.00	**8.58**	**8.58**	**8.58** $\pm$ 0.00	**8.58**
46	**8.07**	**8.07** $\pm$ 0.00	**8.07**	**8.07**	**8.07** $\pm$ 0.00	**8.07**
47	**8.18**	**8.18** $\pm$ 0.00	**8.18**	**8.18**	**8.18** $\pm$ 0.00	**8.18**
48	**8.22**	**8.22** $\pm$ 0.00	**8.22**	**8.22**	**8.22** $\pm$ 0.00	**8.22**
49	**8.82**	**8.82** $\pm$ 0.00	**8.82**	**8.82**	**8.82** $\pm$ 0.00	**8.82**
50	**8.16**	**8.16** $\pm$ 0.00	**8.16**	**8.16**	**8.16** $\pm$ 0.00	**8.16**
51	**8.50**	8.24 $\pm$ 0.24	8.07	**8.50**	8.37 $\pm$ 0.19	**8.50**
52	**8.14**	8.04 $\pm$ 0.17	**8.14**	**8.14**	8.10 $\pm$ 0.11	**8.14**
53	**8.26**	8.11 $\pm$ 0.18	**8.26**	**8.26**	8.20 $\pm$ 0.13	**8.26**
54	**8.33**	8.04 $\pm$ 0.31	7.96	**8.33**	8.30 $\pm$ 0.12	**8.33**
55	**8.84**	8.11 $\pm$ 0.46	7.91	**8.84**	8.72 $\pm$ 0.30	**8.84**
56	**8.16**	8.01 $\pm$ 0.15	8.00	**8.16**	8.12 $\pm$ 0.07	**8.16**
57	**8.64**	7.93 $\pm$ 0.38	7.96	**8.64**	8.01 $\pm$ 0.25	7.96
58	**8.53**	7.95 $\pm$ 0.35	7.89	**8.53**	8.34 $\pm$ 0.23	**8.53**
59	**8.49**	8.00 $\pm$ 0.36	7.87	**8.49**	8.32 $\pm$ 0.25	**8.49**
60	**8.25**	7.82 $\pm$ 0.30	7.82	**8.25**	8.11 $\pm$ 0.19	**8.25**

for large instances.

In general, the best overall results are obtained by MA_{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

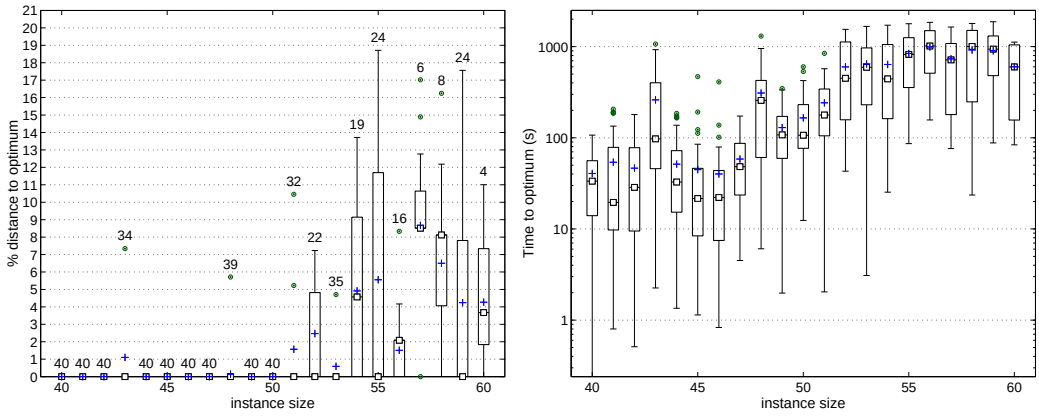


Fig. 9. Relative distance to the optimum (left) and time to find the optimum in seconds (right) for $\mathrm{MA}_{\mathrm{SDLS}}$ and different instance sizes.

for the TS algorithm presented in [22] and the KL algorithm described in [23], and recall that execution platforms are different. In [22], Dotú and Van Hentenryck provided the mean time to find the optimum with respect all the runs, whereas Brglez *et al.* reported in [23] the allowed execution time for each run. Due to this difference, we compared the execution times of $\mathrm{MA}_{\mathrm{TS}}$ with those informed in [22] and [23] separately. Compared to the TS algorithm presented in [22], Table 3 shows that, like the TS algorithm, $\mathrm{MA}_{\mathrm{TS}}$ also finds optimal solutions in all the runs for $L \leqslant 48$ but this is done in a mean time lower than 55 seconds in the worse case (i.e., $L = 48$); in fact, $\mathrm{MA}_{\mathrm{TS}}$ is between 5 (for some instances $L < 40$) and 95 times faster than Dotú and Van Hentenryck's TS without taking into account the smaller computation capacity of our platform, (recall that our PC is about 20% slower than theirs). Compared to [23], the KL algorithm finds the optimum for the $L = 60$ instance in 20 hours of execution time, whereas $\mathrm{MA}_{\mathrm{TS}}$ needs a mean time of 916 seconds, which implies a speed up of about one order of magnitude when the computation power of the different platforms are adjusted (for this comparison, we have estimated that our platform is 900% faster). Moreover, Table 4 shows the allowed execution time (in seconds) for both KL and $\mathrm{MA}_{\mathrm{TS}}$ algorithms when $L \geq 48$: observe again the large differences in allowed execution times for both algorithms.

Also, as shown in Tables 3 and 4, the $\mathrm{MA}_{\mathrm{TS}}$ 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 almost all the runs). In addition, for the larger instances ($L \geq 56$), $\mathrm{MA}_{\mathrm{TS}}$ achieves optimal solutions in most executions (i.e., topping 80%), except for $L \in \{57, 58\}$, for which the success ratios are 48% and 60% (in these cases, the mean distances to the optimum are 4.34% and 1.78%). Observe however that in these higher instances the relation *success/time* is clearly favorable to $\mathrm{MA}_{\mathrm{TS}}$. For example, assuming a number of independent runs of $\mathrm{MA}_{\mathrm{TS}}$ summing up the same computational time (adjusted for platform differences) than one run of KL, we can compute the equivalent success ratios of $\mathrm{MA}_{\mathrm{TS}}$ for sizes from 57 to 60

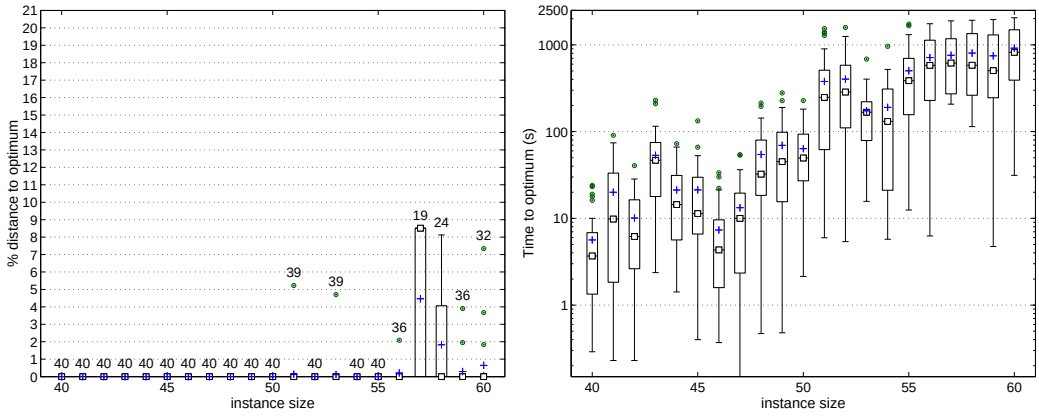


Fig. 10. Relative distance to the optimum (left) and time to find the optimum in seconds (right) for $\mathrm{MA}_{\mathrm{TS}}$ and different instance sizes.

Table 2
Numerical distribution of merit factors for $\mathrm{MA}_{\mathrm{SDLS}}$ and $\mathrm{MA}_{\mathrm{TS}}$ on different instance sizes. Optimal solutions are shown in boldface.

	$\mathrm{MA}_{\mathrm{SDLS}}$			$\mathrm{MA}_{\mathrm{TS}}$		
Instance	Best	Mean $\pm$ σ	Median	Best	Mean $\pm$ σ	Median
40	**7.40**	**7.40** $\pm$ 0.00	**7.40**	**7.40**	**7.40** $\pm$ 0.00	**7.40**
41	**7.78**	**7.78** $\pm$ 0.00	**7.78**	**7.78**	**7.78** $\pm$ 0.00	**7.78**
42	**8.73**	**8.73** $\pm$ 0.00	**8.73**	**8.73**	**8.73** $\pm$ 0.00	**8.73**
43	**8.48**	8.39 $\pm$ 0.20	**8.48**	**8.48**	**8.48** $\pm$ 0.00	**8.48**
44	**7.93**	**7.93** $\pm$ 0.00	**7.93**	**7.93**	**7.93** $\pm$ 0.00	**7.93**
45	**8.58**	**8.58** $\pm$ 0.00	**8.58**	**8.58**	**8.58** $\pm$ 0.00	**8.58**
46	**8.07**	**8.07** $\pm$ 0.00	**8.07**	**8.07**	**8.07** $\pm$ 0.00	**8.07**
47	**8.18**	**8.18** $\pm$ 0.00	**8.18**	**8.18**	**8.18** $\pm$ 0.00	**8.18**
48	**8.22**	8.21 $\pm$ 0.06	**8.22**	**8.22**	**8.22** $\pm$ 0.00	**8.22**
49	**8.82**	**8.82** $\pm$ 0.00	**8.82**	**8.82**	**8.82** $\pm$ 0.00	**8.82**
50	**8.16**	**8.16** $\pm$ 0.00	**8.16**	**8.16**	**8.16** $\pm$ 0.00	**8.16**
51	**8.50**	8.37 $\pm$ 0.25	**8.50**	**8.50**	8.48 $\pm$ 0.06	**8.50**
52	**8.14**	7.95 $\pm$ 0.21	**8.14**	**8.14**	**8.14** $\pm$ 0.00	**8.14**
53	**8.26**	8.21 $\pm$ 0.12	**8.26**	**8.26**	8.25 $\pm$ 0.05	**8.26**
54	**8.33**	7.96 $\pm$ 0.40	7.96	**8.33**	**8.33** $\pm$ 0.00	**8.33**
55	**8.84**	8.41 $\pm$ 0.53	**8.84**	**8.84**	**8.84** $\pm$ 0.00	**8.84**
56	**8.16**	8.04 $\pm$ 0.12	8.00	**8.16**	8.15 $\pm$ 0.04	**8.16**
57	**8.64**	7.96 $\pm$ 0.33	7.96	**8.64**	8.28 $\pm$ 0.33	7.96
58	**8.53**	8.02 $\pm$ 0.32	7.89	**8.53**	8.38 $\pm$ 0.19	**8.53**
59	**8.49**	8.17 $\pm$ 0.44	**8.49**	**8.49**	8.46 $\pm$ 0.07	**8.49**
60	**8.25**	7.92 $\pm$ 0.22	7.96	**8.25**	8.20 $\pm$ 0.11	**8.25**

as 83%, 84%, 99.8% and 99.8% respectively. This way, $\mathrm{MA}_{\mathrm{TS}}$ can be shown to perform at the state-of-the-art level for the LABS problem.

Fig. 11 shows the average evolution along time of the best solution found by the different algorithms (as its relative distance to the optimum) for LABS(55) and LABS(60) instances. As we can observe, $\mathrm{MA}_{\mathrm{TS}}$ dominates the remaining

Table 3
Comparison of MA_{TS} and Tabu Search algorithm in [22]. Table shows mean time in seconds to find the optimum and percentage of success for both algorithms.

	TS [22]		MA_{TS}	
Instance	Time (secs.)	% Success	Time (secs.)	% Success
40	260.11	100	5.65	100
41	460.26	100	20.02	100
42	466.73	100	10.10	100
43	1600.63	100	53.32	100
44	764.66	100	21.23	100
45	1103.48	100	21.26	100
46	703.32	100	7.34	100
47	1005.03	100	13.28	100
48	964.13	100	54.51	100

Table 4
Comparison of MA_{TS} and KL algorithm in [23]. Table shows allowed execution time in seconds and percentage of success for both algorithms.

	KL [23]		MA_{TS}	
Instance	Time (secs.)	% Success	Time (secs.)	% Success
48	1080	68	1380	100
49	1440	75	1440	100
50	2160	93	1500	100
51	2880	31	1560	97
52	4320	75	1620	100
53	6120	75	1680	97
54	8640	62	1740	100
55	12600	87	1800	100
56	18000	100	1860	90
57	47520	68	1920	48
58	35280	81	1980	60
59	50040	100	2040	90
60	72000	100	2100	80

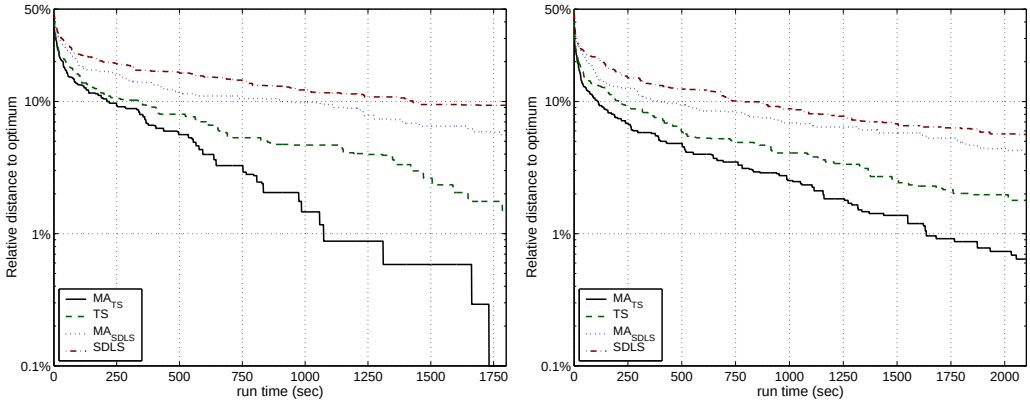
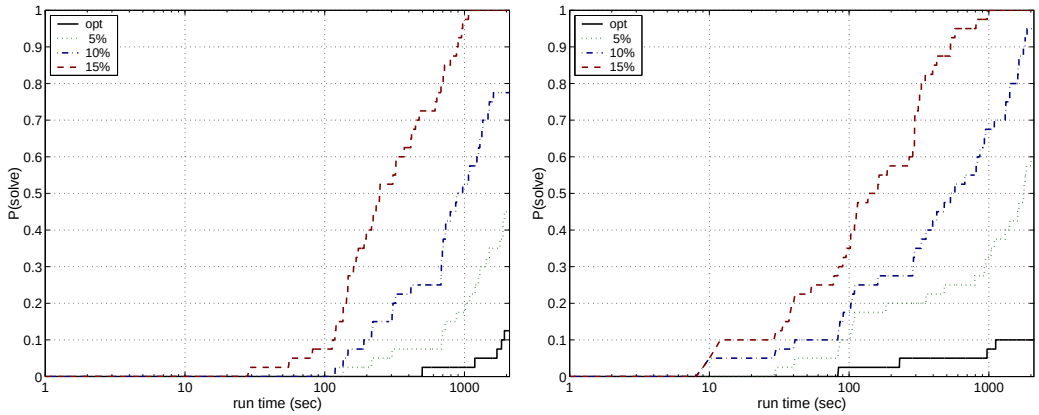
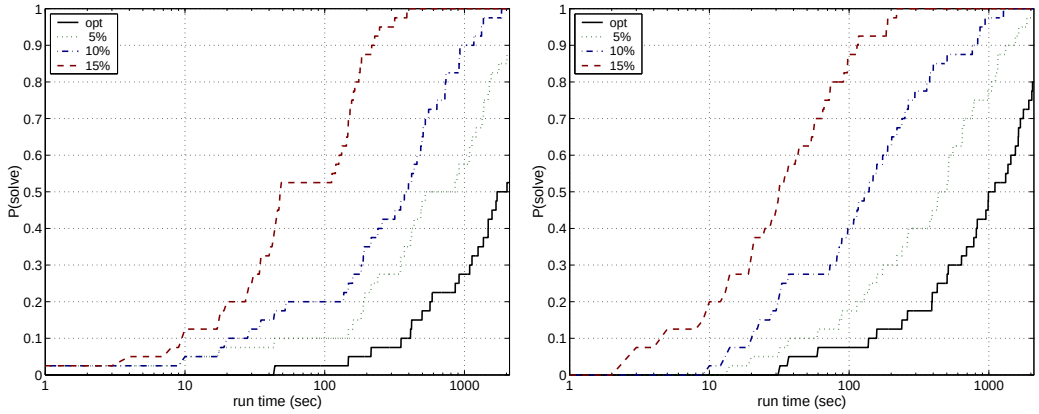


Fig. 11. Evolution along time of the best solution for the different algorithms for LABS(55) (left) and LABS(60) (right). All curves are averaged for 40 runs.

algorithms, as its average solution is consistently better during the whole execution time, followed by TS. This is also the case for all instances in $50 \leq L \leq 60$. We can also observe that MA_{SDLS} performs regularly better for these instances than SDLS alone, a result that holds for all instances in $55 \leq L \leq 60$, showing the benefit of embedding the local search operator within a MA. In this sense, a Wilcoxon rank sum test [24] –also known as Mann-Whitney U test– at the standard 5% significance level confirms that differences of MA_{SDLS} with respect to SDLS are significant for $L \in \{51, 52, 53, 55, 59\}$. The same test indicates that differences of MA_{TS} with respect to TS are always significant for $L > 50$ except for $L \in \{54, 56, 58\}$.



(a) SDLS (b) MA_{SDLS}



(c) TS (d) MA_{TS}

Fig. 12. Qualified run–time distributions for the different algorithms and LABS(60) instance. Figures represent the cumulative probability along time of finding a solution within the specified relative distance to the optimum.

Another graphical representation of the temporal behavior of the different algorithms is provided by Fig. 12, that shows *qualified run–time distributions* (QRTDs) [25] for the LABS(60) instance. Each figure shows the cumulative probability along time of finding a solution within different relative distances to the optimum for a respective algorithm. As it can be seen, TS and MA_{TS} behave in a desirable way, as the probability of finding an optimal solution increases consistently with time. This indicates that these algorithms are not

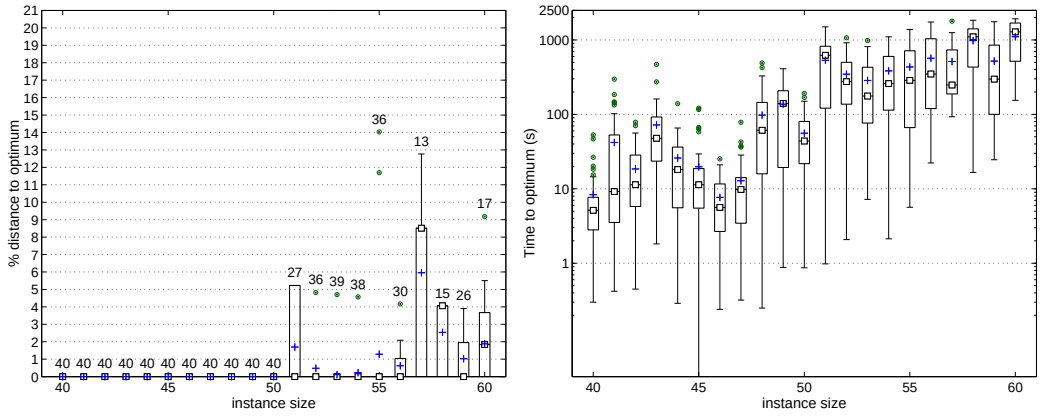


Fig. 13. Relative distance to the optimum (left) and time to find the optimum in seconds (right) for $\mathrm{MA}_{\mathrm{TS\ SPX}}$ and different instance sizes.

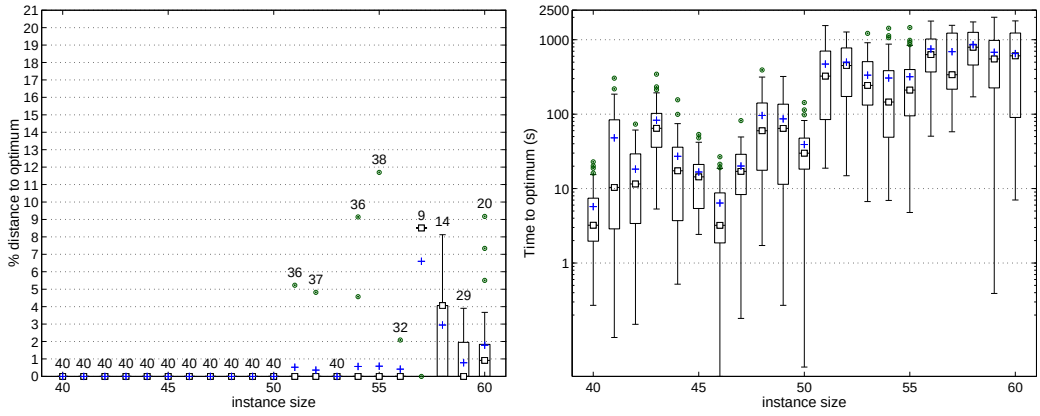


Fig. 14. Relative distance to the optimum (left) and time to find the optimum in seconds (right) for $\mathrm{MA}_{\mathrm{TS\ DPX}}$ and different instance sizes.

stagnated. Both algorithms start producing optimal solutions at about the same time, but the optimal solution curve for $\mathrm{MA}_{\mathrm{TS}}$ is steeper than the one for TS alone, indicating a faster progress of the former algorithm. Note also that QRTDs for tighter quality bounds are shifted to the right as expected, and that the run time required for finding optimal solutions is lower for $\mathrm{MA}_{\mathrm{TS}}$. On the other hand, SDLS and $\mathrm{MA}_{\mathrm{SDLS}}$ start producing solutions latter and the pattern of the corresponding curves for optimal solutions can not be determined within the allowed execution time.

4.2 Other Recombination Operators

$\mathrm{MA}_{\mathrm{TS}}$ utilizes an uniform crossover operator. In this section, we present the results of experiments carried out in order to asses the performance of two different standard operators, namely single point crossover ($\mathrm{MA}_{\mathrm{TS\ SPX}}$) and double point crossover ($\mathrm{MA}_{\mathrm{TS\ DPX}}$). Figs. 13 and 14, and Table 5 show the

distributions for these experiments. The performance of these operators in this problem is generally worse than the one obtained by uniform crossover (e.g., check the success rates for large values of L). This may be due to the high epistasis of the LABS problem. These new operators try to exploit building blocks in solutions, but for the LABS problem all bits in a solution are correlated. In this case, combining elements of the solutions in a uniform manner works better, since this tends to diversify the search, providing new promising starting points for the local searcher.

Table 5
Numerical distribution of merit factors for $MA_{TS\ SPX}$ and $MA_{TS\ DPX}$ on different instance sizes. Optimal solutions are shown in boldface.

	$MA_{TS\ SPX}$			$MA_{TS\ DPX}$		
Instance	Best	Mean $\pm$ σ	Median	Best	Mean $\pm$ σ	Median
40	**7.40**	**7.40** $\pm$ 0.00	**7.40**	**7.40**	**7.40** $\pm$ 0.00	**7.40**
41	**7.78**	**7.78** $\pm$ 0.00	**7.78**	**7.78**	**7.78** $\pm$ 0.00	**7.78**
42	**8.73**	**8.73** $\pm$ 0.00	**8.73**	**8.73**	**8.73** $\pm$ 0.00	**8.73**
43	**8.48**	**8.48** $\pm$ 0.00	**8.48**	**8.48**	**8.48** $\pm$ 0.00	**8.48**
44	**7.93**	**7.93** $\pm$ 0.00	**7.93**	**7.93**	**7.93** $\pm$ 0.00	**7.93**
45	**8.58**	**8.58** $\pm$ 0.00	**8.58**	**8.58**	**8.58** $\pm$ 0.00	**8.58**
46	**8.07**	**8.07** $\pm$ 0.00	**8.07**	**8.07**	**8.07** $\pm$ 0.00	**8.07**
47	**8.18**	**8.18** $\pm$ 0.00	**8.18**	**8.18**	**8.18** $\pm$ 0.00	**8.18**
48	**8.22**	**8.22** $\pm$ 0.00	**8.22**	**8.22**	**8.22** $\pm$ 0.00	**8.22**
49	**8.82**	**8.82** $\pm$ 0.00	**8.82**	**8.82**	**8.82** $\pm$ 0.00	**8.82**
50	**8.16**	**8.16** $\pm$ 0.00	**8.16**	**8.16**	**8.16** $\pm$ 0.00	**8.16**
51	**8.50**	8.36 $\pm$ 0.19	**8.50**	**8.50**	8.45 $\pm$ 0.12	**8.50**
52	**8.14**	8.10 $\pm$ 0.11	**8.14**	**8.14**	8.11 $\pm$ 0.09	**8.14**
53	**8.26**	8.25 $\pm$ 0.05	**8.26**	**8.26**	**8.26** $\pm$ 0.00	**8.26**
54	**8.33**	8.31 $\pm$ 0.07	**8.33**	**8.33**	8.28 $\pm$ 0.14	**8.33**
55	**8.84**	8.74 $\pm$ 0.30	**8.84**	**8.84**	8.79 $\pm$ 0.20	**8.84**
56	**8.16**	8.11 $\pm$ 0.09	**8.16**	**8.16**	8.13 $\pm$ 0.06	**8.16**
57	**8.64**	8.16 $\pm$ 0.33	7.96	**8.64**	8.11 $\pm$ 0.28	7.96
58	**8.53**	8.32 $\pm$ 0.16	8.20	**8.53**	8.29 $\pm$ 0.19	8.20
59	**8.49**	8.40 $\pm$ 0.12	**8.49**	**8.49**	8.42 $\pm$ 0.11	**8.49**
60	**8.25**	8.10 $\pm$ 0.17	8.10	**8.25**	8.11 $\pm$ 0.18	8.18

4.3 Large Instances

We run MA_{TS} for large instances (odd instance sizes in $73 \leq L \leq 85$), in order to compare it to best-known heuristics for the LABS problem. For each instance size, 40 independent executions were run. Execution times were set as before, so that limits were in the range of 2,880 seconds (for $L = 73$) to 3,600 seconds (for $L = 85$). Table 6 shows the results of these experiments.

Table 7 compares solutions provided by MA_{TS} with best-known merit factors in the literature for large instances. In order to better compare different algo-

Table 6
Merit factors for MA_{TS} and large instance sizes.

Instance	Best	Mean $\pm$ σ	Median
73	8.65	7.56 $\pm$ 0.44	7.48
75	8.45	7.41 $\pm$ 0.35	7.23
77	7.84	7.30 $\pm$ 0.23	7.23
79	8.60	7.21 $\pm$ 0.32	7.14
81	7.81	7.17 $\pm$ 0.22	7.16
83	8.10	7.14 $\pm$ 0.34	7.04
85	7.56	6.98 $\pm$ 0.20	6.95

Table 7
Comparison of best-known merit factors in the literature, MA_{TS} and $MA_{TS\ skew}$ for large instance sizes. Best-known solutions are shown in boldface.

Instance	[26]	[27]	[28]	MA_{TS}	$MA_{TS}{}^{10h}$	$MA_{TS\ skew}$
73	7.49	7.66	7.66	**8.65**	**8.65**	7.66
75	8.25	8.25	8.25	8.45	**8.55**	8.25
77	8.10	**8.28**	**8.28**	7.84	**8.28**	**8.28**
79	7.34	7.67	7.67	**8.60**	**8.60**	7.67
81	7.32	**8.20**	**8.20**	7.81	8.04	**8.20**
83	7.81	**9.14**	**9.14**	8.10	8.10	**9.14**
85	7.03	8.17	8.17	7.56	**8.73**	8.17

rithms, best results are shown in boldface. Note that for these large instances, other approaches search only skew symmetric sequences (as the general case is not affordable for those algorithms) whereas MA_{TS} performs a non restricted search. It can be seen that MA_{TS} provides better solutions for $L \in \{73, 75, 79\}$, demonstrating that exploring the whole set of sequences for these instances, instead of restricting the search to a promising subset, is profitable for this heuristic. Note also that the performance of the algorithm degrades when $L \geq 81$. In next section, algorithm $MA_{TS\ skew}$ that considers only skew symmetrical sequences, will be described in depth, and it will be demonstrated that for those instances better results can be achieved by this latter algorithm. With the aim of observing the performance of MA_{TS} for larger execution times, the algorithm was run on these same instances with a time limit of 10 hours. Results are shown in column $MA_{TS}{}^{10h}$ of Table 7. As it can be seen, MA_{TS} was able to improve further its results in most of the instances, providing a total of four new best-known solutions (all new best-known solutions provided by this work are reproduced in Table 10). In any case, note that the solutions for $L \in \{81, 83\}$ are worse than the ones provided by $MA_{TS\ skew}$, although the execution time for this latter algorithm was more than an order of magnitude lower (see Sect. 5).

Table 8
Merit factors for $MA_{TS\ skew}$ and very large instance sizes.

Instance	Best	Mean ± σ	Median
81	8.20	8.20 ± 0.00	8.20
83	9.14	9.14 ± 0.00	9.14
85	8.17	8.17 ± 0.00	8.17
87	8.39	8.39 ± 0.00	8.39
89	8.18	8.18 ± 0.00	8.18
91	8.68	8.68 ± 0.00	8.68
93	8.61	8.61 ± 0.00	8.61
95	9.42	9.42 ± 0.00	9.42
97	8.78	8.66 ± 0.21	8.78
99	8.49	8.45 ± 0.12	8.49
101	8.82	8.70 ± 0.21	8.82
103	9.56	9.28 ± 0.56	9.56
105	8.89	8.83 ± 0.14	8.89
107	8.46	8.37 ± 0.09	8.36
109	8.97	8.64 ± 0.37	8.46
111	8.97	8.43 ± 0.40	8.34
113	8.49	8.28 ± 0.24	8.31
115	8.88	8.41 ± 0.28	8.26
117	8.71	8.20 ± 0.30	8.13
119	8.48	7.90 ± 0.28	7.81
121	8.67	8.20 ± 0.31	8.06
141	8.83	7.62 ± 0.33	7.52
161	8.57	7.53 ± 0.32	7.45
181	7.72	7.14 ± 0.22	7.10
201	7.66	6.93 ± 0.26	6.84

5 Skew Symmetric Sequences

One subset of sequences that has gained much attention in the search of LABS are so called *skew-symmetric* sequences. These are sequences with odd length L fulfilling

$$s_{n+i} = (-1)^i s_{n-i}, \quad n = (L+1)/2, \quad \text{for } 1 \leq i \leq n-1, \tag{6}$$

from which it follows that $C_k(S) = 0$ for odd values of k. Working with skew-symmetric sequences, reduces by 2^{n-1} the size of the search space that has to be explored (at the expense of not guaranteing optimality for the general case). Although Golay derived the same asymptotic value of F_L for these sequences that for the general case, true global optimum are not skew-symmetric for several values of L (see [17]). Anyway, many heuristics approaches to the LABS problem have restricted search to the subspace of skew symmetric sequences, with the aim of being effective for very large instances of the problem.

In this section, we experiment on adapting MA_{TS} to search in the space of

Table 9
Comparison of best-known merit factors in the literature and $MA_{TS\ skew}$ for very large instance sizes. Best-known solutions are shown in boldface.

Instance	[26]	[27]	[20]	[28]	[14]	$MA_{TS\ skew}$
101	6.06	8.36	8.36	8.36	**8.82**	**8.82**
103	5.90	**9.56**	**9.56**	**9.56**	**9.56**	**9.56**
105	6.07	**8.89**	8.25	**8.89**	8.78	**8.89**
107	6.53	**8.46**	**8.46**	8.36	**8.46**	**8.46**
109	6.15	**8.97**	**8.97**	7.84	**8.97**	**8.97**
111	6.02	**8.97**	**8.97**	7.95	**8.97**	**8.97**
113	6.33	**8.49**	**8.49**	8.31	**8.49**	**8.49**
115	6.40	**8.88**	8.60	7.79	**8.88**	**8.88**
117	6.42	**8.71**	8.12	**8.71**	**8.71**	**8.71**
119	6.01		7.67	7.54	8.02	**8.48**
121	6.61		**8.67**		**8.67**	**8.67**
141	6.01		7.45		**8.83**	**8.83**
161	6.02		6.89		8.39	**8.57**
181	5.70		6.77		**7.75**	7.72
201			6.29		7.46	**7.66**

skew symmetric sequences. To this end, a solution for a LABS(L) instance consists of the first $(L+1)/2$ bits of the sequence. The remaining bits are calculated using Eq. (6) in order to obtain a skew symmetric sequence. Recall that $C_k(S) = 0$ for odd values of k, and hence, these correlations do not have to be computed when calculating the fitness of a skew symmetric sequence. Accordingly, the probability of mutation have been redefined as $p_m = 1/((L+1)/2)$. The MA so obtained will be named $MA_{TS\ skew}$. This algorithm was run on very large instances (odd instance sizes in $81 \leq L \leq 121$) and ($L \in \{141, 161, 181, 201\}$). 40 independent executions were carried out for each instance size. Regarding time limits, we followed the same methodology taken for MA_{TS}, but considering that solving the skew symmetrical case for L is approximately as hard as solving the general case for $L/2$, and, for the first group of instances, a time limit of 900 seconds was imposed for $L = 81$. This limit was increased by 30 seconds with each increment in instance size (i.e. the greatest time limit was 35 minutes for $L = 121$). Clearly, the complexity of the problem does not grow linearly, and hence larger execution times were given to the second group of instances. To be precise, the time limit for $L = 141$ was of one our, and this time limit was incremented in 10 minutes for each size increment. Table 8 shows the results of these experiments. Going back to Table 7, we see that $MA_{TS\ skew}$ performs better than MA_{TS} for $L \in \{81, 83\}$, although for these instances the allowed time is more than one order of magnitude lower. This shows that for these instances, it is more profitable to used $MA_{TS\ skew}$ than MA_{TS}.

On Table 9, $MA_{TS\ skew}$ is compared against best-known methods in the literature. Only the best solutions are compared, as this is the only informa-

Table 10
New best-known solutions provided by different MAs studied in this paper. Solutions are shown in run length encoded notation.

Instance	Solution
73	4732231714112121113112211321132111111
75	3233321322213121162121211211611151111111
79	113111131121313215213211823121121123442
79	111111131112311212143418112112741222222 1
85	131312212131311113121121121411112133211143634
85	111212111121211521213611344316131332313 12
119	113311113113323212115613115123111413111121114321221214161213
161	141112411111161331111124111213222314332112134212121131222213 51127121311118112511 2
201	114171412111312111232311112311121121414614225223221222111222 113111121131134512612212531531131111131 13

tion provided in the literature for other approaches. $\text{MA}_{\text{TS skew}}$ always (except for $L = 181$) provides the best solution. Note that for three instances ($L \in \{119, 161, 201\}$) it produces new best-known solutions.

Regarding execution times, it is difficult to compare $\text{MA}_{\text{TS skew}}$ to other approaches, as the available information is limited. For instance, in [28], the only information provided is that their algorithm was tested on skew symmetrical sequences with $73 \leq L \leq 119$, and that it "was allowed one run of 10^9 steps per problem, each run taking several hours, with a small number of additional runs when results were poor." These experiments were carried out on a 733 MHz Pentium III PC. In any case, note that the number of individuals generated by $\text{MA}_{\text{TS skew}}$ was less than 10^6 for the same range of instances, with a maximum execution time of 34 minutes on a 2.4 GHz Pentium IV PC for $L = 119$. With respect to the (μ, λ)-ES of [14], it is reported that their results were obtained by running the algorithm for 2×10^6 generations, with $\mu = 10$ and $\lambda \approx L$, which implies a total of $2 \times L \times 10^6$ generated individuals. It is also remarked that, "due to the nondeterministic character of the ES, the size of the search space and the roughness of the fitness function, the results strongly depend on the initial conditions, making it unlikely to reobtain the best values in every run". In order to compare this ES to $\text{MA}_{\text{TS skew}}$, note that the complexity of their mutation operator is bounded by $L^2/2$, whereas the complexity of the local search operator used in $\text{MA}_{\text{TS skew}}$ is bounded by $2(L/2)^3/3$. As the number of generations in each execution for $\text{MA}_{\text{TS skew}}$is less than 10^6 when $81 \leq L \leq 121$, the total number of operations performed for this latter algorithm is more than an order of magnitude lower than the ones performed by the ES for these instances. For $L > 121$, the number of generations in each execution for $\text{MA}_{\text{TS skew}}$is less than 3×10^6, and hence the comparison is again favorable to this algorithm.

6 Conclusions

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 substantially improved.

We have also applied the MA_{TS} algorithm to larger LABS instances for which the optimal is unknown, in order to test the scalability of the approach in the long term. Results show that the algorithm is capable of systematically recovering best know solutions and is even able to provide some new best-known solutions. Additionally, we have adapted the algorithm to explore only skew-symmetric solutions. This allows testing the algorithm on even larger instances. Results show that this variant of the algorithm produces new best-known results for very large instances.

Acknowledgements

This work was partially supported by Spanish MICINN under contracts TIN2004-7943-C04-01 and TIN2008-05941.

References

[1] R. J. Turyn, J. Storer, On binary sequences, Proceedings of the American Mathematical Society 12 (1) (1961) 394–399.

[2] R. J. Turyn, Sequences with small correlation, in: H. B. Mann (Ed.), Error Correcting Codes, Wiley, New York, 1968, pp. 195–228.

[3] J. Bernasconi, Low autocorrelation binary sequences: statistical mechanics and configuration state analysis, Journal de Physique 48 (4) (1987) 559–567.

[4] P. Stadler, Landscapes and their correlation functions, Journal of Mathematical Chemistry 20 (1) (1996) 1–45.

[5] V. de Oliveira, J. Fontanari, P. Stadler, Metastable states in high order short-range spin glasses, Journal of Physics A: Mathematical and General 32 (1999) 8793–8802.

[6] J. E. Gallardo, C. Cotta, A. J. Fernández, A memetic algorithm for the low autocorrelation binary sequence problem, in: D. Thierens, et al. (Eds.), 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), ACM Press, New York, USA, 2007, pp. 1226–1233.

[7] P. Moscato, Memetic algorithms: A short introduction, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999, pp. 219–234.

[8] P. Moscato, C. Cotta, A gentle introduction to memetic algorithms, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, Boston MA, 2003, pp. 105–144.

[9] W. E. Hart, N. Krasnogor, J. E. Smith, Recent Advances in Memetic Algorithms, Vol. 166 of Studies in Fuzziness and Soft Computing, Springer-Verlag, Berlin Heidelberg, 2005.

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

[11] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, 1997.

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

[13] M. J. E. Golay, Sieves for low autocorrelation binary sequences, IEEE Transactions on Information Theory 23 (1) (1977) 43–51.

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

[15] F. Ferreira, J. Fontanari, P. Stadler, Landscape statistics of the low autocorrelated binary string problem, Journal of Physics A: Mathematical and General 33 (2000) 8635–8647.

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

[17] S. Mertens, H. Bauke, Ground states of the Bernasconi model with open boundary conditions, website available at *http://www-e.uni-magdeburg.de/mertens/research/labs/open.dat* (accessed January 2007).

[18] Q. Wang, Optimization by simulating molecular evolution, Biological Cybernetics 57 (1987) 95–101.

[19] C. de Groot, D.Würtz, K. Hoffmann, Low autocorrelation binary sequences: exact enumeration and optimization by evolutionary strategies, Optimization 23 (4) (1992) 369–384.

[20] A. Reinholz, Ein paralleler genetischer algorithmus zur optimierung der binarien autokorrelations-funktion, Diploma Thesis. Universität Bonn (October 1993).

[21] S. Prestwich, A hybrid local search for low autocorrelation binary sequences, Technical Report TR-00-01, Department of Computer Science, National University of Ireland, Cork, Ireland (2000).

[22] I. Dotú, P. V. Hentenryck, A note on low autocorrelation binary sequences, in: F. Benhamou (Ed.), 12th International Conference on Principles and Practice of Constraint Programming (CP 2006), Vol. 4204 of Lecture Notes in Computer Science, Springer-Verlag, Nantes, France, 2006, pp. 685–689.

[23] F. Brglez, X. Y. Li, M. F. Stallman, B. Militzer, Reliable cost prediction for finding optimal solutions to LABS problem: Evolutionary and alternative algorithms, in: E. Cantú-Paz (Ed.), 5th International Workshop on Frontiers in Evolutionary Algorithms (FEA 2003), Cary, North Carolina, USA, 2003.

[24] E. L. Lehmann, Nonparametric Statistical Methods Based on Ranks, McGraw-Hill, New York, 1975.

[25] H. Hoos, T. Stutzle, Stochastic Local Search. Foundations and Applications, Morgan Kaufmann Publishers, San Francisco, 2005.

[26] G. F. M. Beenker, T. A. C. M. Claasen, P. W. C. Hermens, Binary sequences with a maximally flat amplitude spectrum, Philips Journal of Research 40 (5) (1985) 289–304.

[27] M. J. E. Golay, D. B. Harris, A new search for skewsymmetric binary sequences with optimal merit factors, IEEE Transactions on Information Theory 36 (5) (1990) 1163–1166.

[28] S. Prestwich, Exploiting relaxation in local search, in: J. Pearson, M. Bohlin, M. Ågren (Eds.), 1st International Workshop on Local Search Techniques in Constraint Satisfaction (LSCS 2004), Toronto, Canada, 2004, pp. 49–61.