Preprint of the accepted submission to
Information Sciences,
to be published in 2004.

Evolutionary and Alternative Algorithms: Reliable Cost Predictions for Finding Optimal Solutions to the LABS Problem*

Franc Brglez[a,1], Xiao Yu Li[a], Matthias F. Stallmann[a], Burkhard Militzer[b]

[a]Computer Science Department, NC State University, Raleigh, NC 27695, USA

[b]Carnegie Institution of Washington, Washington, DC 20015, USA

Abstract

The *low-autocorrelation binary sequence* (LABS) problem represents a major challenge to all search algorithms, with the evolutionary algorithms claiming the best results so far. However, the termination criteria for these types of stochastic algorithms are not well-defined and no reliable claims have been made about optimality. Our approach to find the optima of the LABS problem is based on combining three principles into a single method: (1) solver performance experiments with problem sizes for which optimal solutions are known, (2) an asymptotic statistical analysis of such experiments, (3) reliable predictions of the computational cost required to find optimal solutions for larger problem sizes. The proposed methodology provides a well-defined termination criterion for evolutionary and alternative search algorithms alike.

keywords: fundamental combinatorial algorithms, performance predictions and evaluations, low-autocorrelation binary sequence, evolutionary algorithms

*Ready-to-execute software environment, data sets, and the results reported in the experiments have been archived on the LABS problem home page `www.cbl.ncsu.edu/OpenExperiments/LABS/`

Email adddresses:
brglez@cs.ncsu.edu, xyli@unity.ncsu.edu, stallmann@cs.ncsu.edu, militzer@gl.ciw.edu

[1]Corresponding author.

1 Introduction

The *low-autocorrelation binary sequence* (*LABS*) problem has a simple formulation. Consider a binary sequence of length L, $S = s_1 s_2 \dots s_L$ and their autocorrelations $C_k(S) = \sum_{i=1}^{L-k} s_i s_{i+k}$, $s_i = \{+1, -1\}$, for $k = 1, \dots, L-1$. The energy function of interest is

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

which defines the *merit factor* F of the sequence [1]:

$$F = L^2/(2E). \tag{2}$$

The objective of optimization is to find a sequence that maximizes F, or equivalently, minimizes E. Finding an optimum sequence has important applications in communication engineering and is also of interest to physicists since the sequence models one-dimensional systems of Ising-spins. The solution of the LABS problem corresponds to the ground state of a generalized one-dimensional Ising spin system [2]. The problem as formulated in (1) is NP-hard or worse, unlike the special cases of the Ising spin-glass problems with limited interaction and periodic boundary conditions evaluated in [3, 4]. While efficient and effective methods have been presented to solve the special cases up to $L = 400$ [3, 4], the optimal merit factors for the problem as formulated in (1) are presently known for values of $L \leq 60$ only [5, 6]. The difficulty of the LABS problem arises not only from the inconsistency or frustration of different energy interaction terms [7] but also from the fact that for most values of L, the number of global optima is relatively small, as shown in Table 1. The results as reported in Table 1 are a by-product of the approach we introduce in this paper.

The asymptotic value for the maximum merit factor F is known [1] and has also been re-derived using arguments from statistical mechanics [2]: as $L \to \infty$, $F \to 12.3248$. In Figure 1, we plot the currently known values of the merit factor as the function of L, *normalized* with respect to 12.3248. The values for $L \leq 60$ are based on a branch-and-bound solver and are thus known to be optimal [5, 6]. For $L > 60$, the values are based on the best known merit factors reported by stochastic search solvers [8, 9, 10, 11, 12], and also on the work in this paper. It shoud be noted that these results are significantly better than the solutions based on simulated annealing in [2].

Table 1: The number of optimal sequences for the LABS problem. The values listed for $L \leq 35$ are exact. For $L > 35$, the values are based on our experiments to date. Some are rounded to the nearest integer multiple of 4. For odd values of L, the number of optimal sequences that are skew-symmetric is shown between the brackets.

L	0	1	2	3	4	5	6	7	8	9
				4(4)	8	4(4)	28	4(4)	16	24(8)
10	40	4(4)	16	4(4)	72	8(8)	32	44(4)	16	8(0)
20	8	4(4)	24	24(0)	8	8(0)	24	4(4)	8	8(8)
30	16	8(0)	8	8(0)	8	8(0)	8	8(0)	8	8(8)
40	8	4(4)	8	4(4)	8	4(4)	24	28(20)	8	12(4)
50	12	4(4)	8	8(4)	8	8(8)	8	4(4)	8	12(4)
60	12	8(0)	12	4(0)	4					

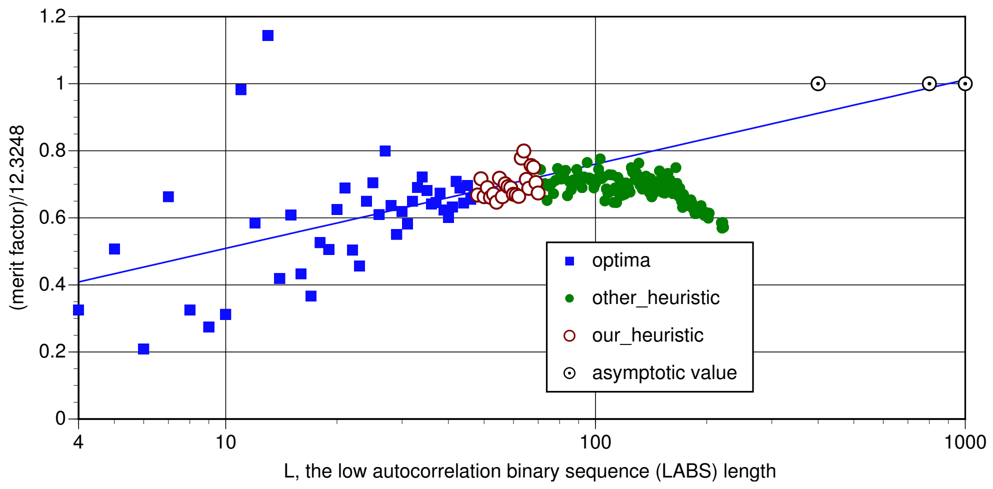


Figure 1: Exact optima and the best known figures of merit for the low autocorrelation binary sequences, normalized to the asympotic limit of 12.3248. The exact optima are known only for $L \leq 60$. The regression line is based on the least-square fit to the data points shown. There clearly is a significant 'gap of opportunity' for the new generation of LABS solvers, starting at $L > 100$.

The simple formulation and the known asymptotic value for large values of L makes the LABS problem a particularly interesting case of a problem that is most likely NP-hard or worse. As depicted in Figure 1, the problem presents a significant challenge not only to the branch-and-bound solvers but also to the stochastic solvers: while the exact figures of merit, limited to values of $L \leq 60$, follow the expected trend, the figures of merit reported by current heuristics for values of $L > 100$ are clearly diverging from the expected trend. Questions that arise include: (1) are the current stochastic solvers inadequate for the task, (2) are the solvers being terminated prematurely, and (3) is the current trend due to some combination of (1) and (2). We argue that the most likely answer is (3) and that the problems for $L > 100$ may be solved better with improved solver technology. Consider, for example, the advances in solving problems in satisfiability (SAT). Real-world SAT problems with 100s of decision variables can today be solved routinely and reliably using branch-and-bound SAT solvers as well as stochastic SAT solvers [13].

Our approach to finding the optima of the LABS problem is based on combining three principles into a single method: (1) solver performance experiments with problem sizes for which optimal solutions are known, (2) an asymptotic statistical analysis of such experiments, (3) reliable predictions of the computational cost required to find optimal solutions for larger problem sizes. Two factors motivated the inclusion of an evolutionary search (ES) solver in this work: (1) the best results for $L > 100$ reported to date in the open literature are based on an ES-based LABS solver [10], and (2) the predictions of computational cost with our version of the LABS solver, which made our solver appear too expensive [14] for $L > 100$. It should be noted that while it was obvious from our experiments that our solver is highly efficient computation-wise when compared to the branch-and-bound solver as reported in [5] (for values of $L \leq 60$), we could not conclude one way or the other with respect to the ES-solver – until both solvers were installed on the same platform and under the same termination criteria. Subsequently, we demonstrate in the paper that the proposed methodology provides a well-defined termination criterion for evolutionary and alternative search algorithms alike.

The paper expands on our work as reported in [15] and is organized as follows. Section 2 briefly introduces two very different LABS solvers and the termination criteria that we apply to both, Section 3 defines the experimental set up that remains identical for both solvers, Section 4 summarizes the asymptotic performance of both solvers under *termination criterion 'A'* for $L \leq 47$, Section 5 summarizes the predictions and the asymptotic performance of the LABS solver of choice under *termination criterion 'B'* for $L > 47$, Section 6 addresses few questions that arise in view of the reported experimental results, Section 7 outlines conclusions and future work.

2 LABS Solvers and Termination Criteria

We introduce two LABS solvers at a level that will support a thorough one-on-one comparative analysis for both solvers. Evolutionary search (ES) techniques introduced in [16, 17] provide the background for the ES-solver that has been devised and optimized explicitly for the LABS problem [10]. Our alternative LABS solver, the KL-solver, is based on principles introduced in the classical graph partitioning paper by Kernighan and Lin [18] and subsequently adapted to hypercube embedding [19, 20].

2.1 ES Solver

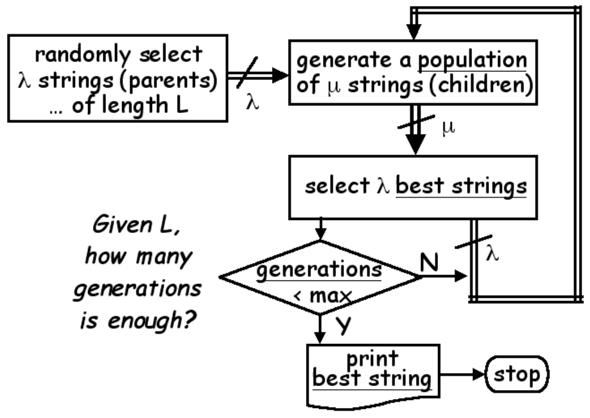
Evolutionary strategies (ES) in its standard form start with an initial parent population of λ binary strings, each of length L, and generated at random. Then μ children strings are generated, each by selecting a parent at random and applying a randomly chosen mutation operation to it. From the μ children strings, one selects the λ best individuals, which form the next parent population. The process is repeated until a termination criterion is met. Termination criteria are discussed later in this section. Militzer et al. [10] developed a novel mutation operator that appears to work particularly well for the LABS problem. A flowchart view of the ES-solver as it traces the LABS problem landscape (for $L = 7$) is shown in Figure 2. Clearly, with a single decision point, its 'control structure' at the level shown is very simple – compared to the control structure at the same level for the KL-solver, to be discussed shortly.

The main objectives of showing the traces below the ES flowchart are the following: (1) even for $L = 7$, the energy cost (or fitness) function can vary two-orders of magnitude; (2) for choices of $\lambda = 3$ and and $\mu = 5$, and running the solver for 4 generations (a total of 23 samples) returns a local optimum solution with the energy cost of 7; (3) for choices of $\lambda = 3$ and and $\mu = 5$, and running the solver for 9 generations (a total of 48 samples) returns a global optimum solution with the cost of 3. Of course, the choice of the ES parameters in this example is for simplicity of illustration only; we discuss the actual values used in the section that follows.

2.2 KL Solver

Our implementation of the KL solver samples the LABS cost function in a sequence of well-defined *moves*, starting from a randomly chosen initial binary string and evaluating the distance-1 neighborhood of this string (flipping 1-bit at a time). We mark the bit for which the cost function was minimum and now repeat the process of evaluating the distance-1 neighborhood, but now only with respect to the bits that are left unmarked. After the last

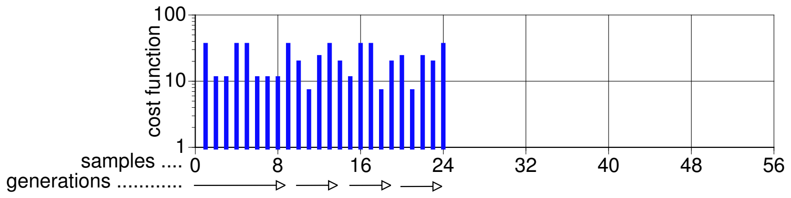
Evolutionary strategy (ES) solver: key procedures and decisions


randomly select λ strings (parents) ... of length L
λ
generate a population of μ strings (children)
μ
select λ best strings
λ
Given L, how many generations is enough?
generations < max
N
Y
print best string
stop

Solving the LABS problem of length $L = 7$:

(a) four generations of population samples find a local optimum cost of 7.

Here, $\lambda = 3, \mu = 5$ and total samples $= 3 + 4 * 5 = 23$.



(b) nine generations of population samples find a global optimum cost of 3.

Here, $\lambda = 3, \mu = 5$ and total samples $= 3 + 9 * 5 = 48$.

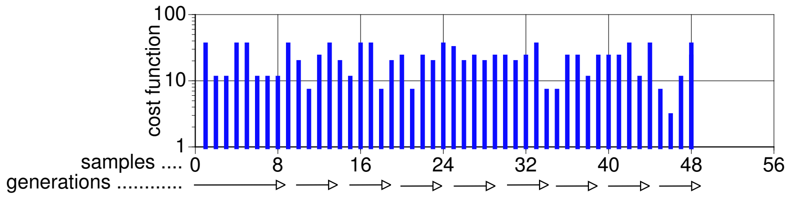
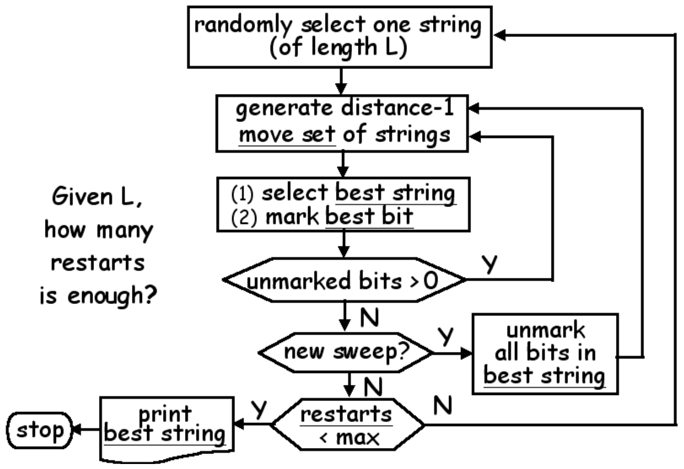


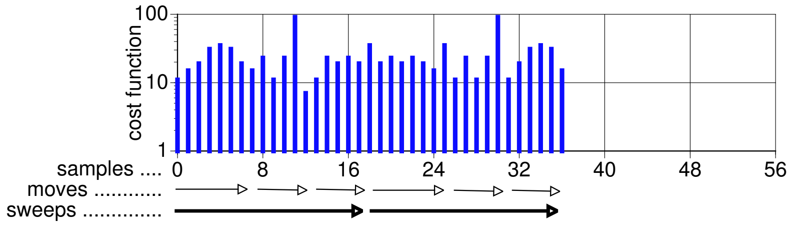
Figure 2: A flowchart view of the ES-solver as it traces the LABS problem landscape.

Kernighan-Lin (KL) solver: key procedures and decisions


randomly select one string (of length L)
generate distance-1 move set of strings
(1) select best string
(2) mark best bit
unmarked bits > 0
Y
N
new sweep?
Y
unmark all bits in best string
N
restarts < max
N
Y
print best string
stop
Given L, how many restarts is enough?

Solving the LABS problem of length $L = 7$:

(a) two sweeps of three move set samples terminate at a local optimum cost of 7. Here, restarts = 0. Total samples $= 1 + 2 * (7 + 6 + 5) = 37$.



(b) three sweeps of three move set samples terminate at a global optimum cost of 3. Here, restarts = 1. Total samples $= 1 + 3 * (7 + 6 + 5) = 55$.

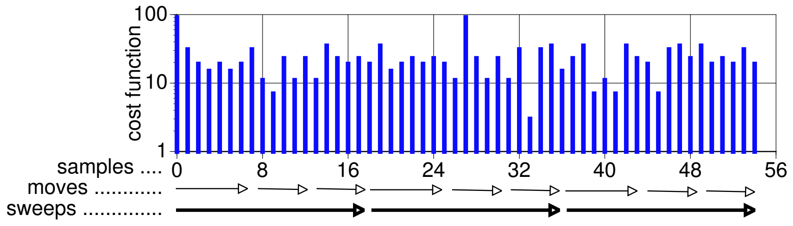


Figure 3: A flowchart view of the KL-solver as it traces the LABS problem landscape.

marking of the bits, we have evaluated $L * (L-1)/2$ samples. We use the 'best string' from this search as the new starting point for the new *sweep* of distance-1 move set (after first unmarking all of the bits). These sweeps are terminated once a new sweep produces a 'best string' that is not better than the one saved from the previous sweep. At this point, the process is repeated with a *new randomly selected string* until a termination criterion is met. Termination criteria are discussed later in this section. A flowchart view of the KL-solver as it traces the LABS problem landscape (for $L = 7$) is shown in Figure 3. Clearly, with three decision points, the 'control structure' at the level shown is not as simple when compared to the control structure at the same level for the ES-solver discussed earlier. For more details about the KL-solver algorithm, see the Appendix.

The main objectives of showing the traces below the KL flowchart are the following: (1) even for $L = 7$, the energy cost (or fitness) function can vary two-orders of magnitude; (2) unlike for ES, no user judgment is required to set-up the parameters that govern the selection of strings. For simplicity of illustration only, we make the decision that 'all bits have been marked' at the value of $7 - 7/2 = 4$, so that each sweep contains only $(7 + 6 + 5) = 18$ samples. Here, we terminate after 0 restarts (a total of 37 samples) and return a local optimum solution with the energy cost of 7; (3) Similarly to (2) above, we terminate after 1 restart (a total of 55 samples) and return a global optimum solution with the energy cost of 3.

2.3 Termination Criteria

The simplest termination criterion in a stochastic search solver is the maxium time duration (timeout) of the experimental run from a given starting point. Such criterion is platform-dependent. However, even if the two LABS solvers such as ES-solver and KL-solver described above are being executed on the same platform under the same timeout limit, we may learn very little about *what factors* may account for the significant difference in the observed performance of the two solvers. Note also that that the flow-chart level, the combinatorial metrics decide the termination for each solver are different: the maximum number of generations (ES) versus the maximum number of restarts (KL). However, if we count the number of *samples*, i.e. cost function evaluations performed by each solver, we do have a metric that is common to both. Experiment show that *samples* correlate perfectly with *runtime* (in seconds) of either solver.

To improve the efficiency and the reliability of the solver selection process, we organize the experiments under two termination criteria [14]:

- Termination criterion 'A' (for $L \leq 47$). Use the known LABS minimum value for a given L to terminate a local search algorithm when the optimal sequence is found for the first time. Variables recorded during this search include, at the minimum, *runtime* and *samples* (the number of cost function evaluations) that are required to find the optimum. Both are random variables for which we find statistics so we can deduce the *probability of success*, P_{succ}, of finding the LABS minimum value, given the *runtime* or *samples* constraints.
- Termination criterion 'B' (for $L > 47$). Develop a predictor model based on results under Termination A to predict, for a given value of L, the *required runtime* and/or *required samples* for which the solver must be allowed to run in order to find the global minimum with the probability of success P_{succ}. The value of *required samples* must be used as the *termination criterion 'B'* if experiments are executed on a platform different from the one used under the *termination criterion 'A'*.

Of course, we will run only the 'best-of-the-two' from KL-solver and ES-solver under the termination B (for the given value of L).

We summarize the details of experimental set-up in the next section. In particular, we demonstrate that experiments under the termination criterion 'A' lead to a number of important insights about the asymptotic and statistical performance of each solver.

3 Experimental Setup

The experimental testbed we use is outlined in Figure 4. Each experiment is initialized with a unique random seed from the file *randomTriplets* that contains three random integers on each line, making the testbed environment cross-platform consistent for any LABS solver that uses the standard IEEE random generator. Under the criterion 'A', the termination of each execution is controlled by the known optimum value stored in the file *knownOptima*, available at [6].

Data of most interest generated by these experiments are *runtime* (seconds, platform-specific) and *samples* (total number of samples to evaluate the cost function, platform-independent). These two random variables are closely correlated and, not surprisingly, both will have *exponential distribution* for both ES and KL solver.[1] While the exponential distribution of *runtime* or *samples* implies significant variability as illustrated in Figure 4, the variability is also *predictable* – shown by the close fit of observed and theoretical distributions. For example, if we allow the KL solver to search for the optimum sequence of length $L = 34$ for 3,088,247 samples (from *any randomly choosen initial sequence*), the probability of finding the optimum is 0.632. However, if we increase the search limit to 4*3,088,247 samples or 8*3,088,247 samples, the probability increases to 0.981 or 0.999 respectively.

The parameters used for all ES-solver experiments are based on settings evaluated as 'best overall' in [10]: $\lambda = 10$ and $\mu = 3 * L$. The single parameter used in KL-solver experiments is *bits-to-mark* $= L/2$. This is different from a default value of *bits-to-mark* $= L$ since we observed no significant improvement (with respect to the LABS problem) if we marked all bits of the *move set* as defined in Figure 3.

Thus, before starting the experiments with the two solvers, we made every effort not to violate the "PET PIEVE 10. *Hand-tuned algorithm parameters*" that concludes with a recommended rule as follows [22]:

> If different parameter settings are to be used for different instances, the adjustment process must be well-defined and algorithmic, the adjustment must be described in the paper, and the time for the adjustment must be included in all reported running times.

We analyze the statistical performance of each solver in the next section.

[1]Exponential distribution of runtime has been observed not only for stochastic solvers in different contexts [21], but also within a unified framework for stochastic *and* branch-and-bound solvers applied to SAT problem instances [13]. Distributions other than exponential have also been observed in a number of related experiments with several solvers [13].

Experiments under termination 'A'

Create *LABSbed* as a LABS-problem specific testbed environment, simpler than but similar to *SATbed* [23]. We consider (1) primary input files *randomTriplets* and *knownOptima*, (2) user-specified parameters L, *solverList*, *nExperiments*, (3) a solver encapsulator SolverEncap_A(L, *randomTriplets*, *knownOptima*, *solverList*, *nExperiments*) that returns *rawData*(*solver*, L), and (4) GetStats(*rawData*(*solver*, L)) that returns *statsData*(*solver*, L).

Invoke SolverEncap_A to return a tabular report *rawData*(*solver*, L) for each solver and each L, with data columns under labels such as instanceNum, runtime, samples, solution. The number of rows is determined by *nExperiments*. For the same value of L, each experiment represents an equivalent problem instance, initialized by the LABS solver with a different random seed triple from the file *randomTriplets*.

Invoke GetStats to return various statistics for random variables in each data column in *rawData*(*solver*, L), including characterization of underlying distributions and the respective solvability functions [23]. As shown below, the number of samples (or function evaluations) required by the KL solver to find the optimum solution for $L = 34$ for *each* of the 128 experiments has exponential distribution and can vary dramatically. At best, an optimum solution is found with 43,354 samples; at worst with 13,322,800 samples; with a median, mean and standard deviation of 2,11,5035 and 3,088,247 and 3,006,546 samples, respectively. The runtime mean value, specific to our platform (a Linux PC with 266 MHz processor), is 6.60 seconds. Hyperlinks to complete raw and statistical results of experiments such as illustrated here, covering several solvers, and values of L up to 64 can be found on the newly created *LABSbed* home page [24].

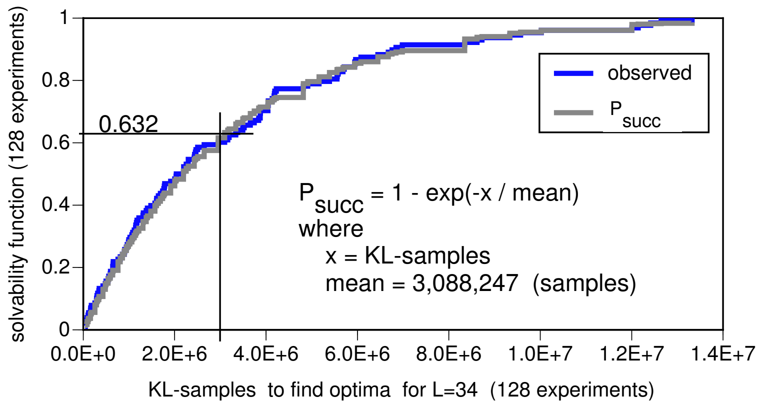


Figure 4: Experimental flow under termination 'A' (for any solver).

4 LABS Solver Comparisons

To capture the asymptotic performance of the ES and KL solver under the termination 'A' with statistical significance, we performed 128 experiments with each solver for each value of L in the range of $10 \le L \le 47$. Similarly to Figure 4, we observe for each L an exponential distribution of *runtime* and *samples* for both solvers. Also, we observe exponential distribution for *generations* (ES solver) and *restarts* (KL solver).[2]. However, with 128 trials the distribution of the reported means for each of these random variables is near-normal due to the law of large numbers. Also with 128 trials, two means with comparable standard deviation will pass a standard t-test for equivalence, at confidence level of 95%, as long as their ratios are bounded by 4/3 from above and 3/4s from below. In other words, after 128 independent experiments, we shall declare two LABS solvers such as ES and KL to have significantly different performance only if the ratio of their means exceeds 4/3 or is less than 3/4. For example, in Figure 4 we report for the value of $L = 34$, the mean number of samples as 2,925,765 (ES solver) and – slightly better than the mean number of samples 3,088,427 (KL solver). A t-test declares both means to be equivalent, and a χ^2-test declares both distributions to be equivalent.

Comparison with a single value of L is not sufficient; we rely on the asymptotic analysis for averages and ratios of averages based on 128 experiments for consecutive values of $L = 10$ to $L = 47$ for both KL and ES solvers. As shown in Figure 5, we pair data reported by the two solvers in terms of average *samples*, average *runtime* (in seconds), and the *sampling efficiency* defined as the ratio of the average number of *samples* to the average *runtime*. For each set of data we also show a regression line that *nominally predicts* the asympotic behavior of each solver beyond the value of $L = 47$:[3]

$$\begin{aligned} \textit{ES\_samples predictor} &= 5.558\text{E+}1 \times 1.370^L && (3)\\ \textit{ES\_runtime predictor} &= 7.108\text{E–}4 \times 1.397^L && (4)\\ \textit{ES\_efficiency predictor} &= 7.819\text{E+}4 \times 0.981^L && (5)\\ \textit{KL\_samples predictor} &= 1.553\text{E+}1 \times 1.423^L && (6)\\ \textit{KL\_runtime predictor} &= 1.287\text{E–}5 \times 1.463^L && (7)\\ \textit{KL\_efficiency predictor} &= 1.206\text{E+}6 \times 0.973^L && (8) \end{aligned}$$

Note however, that not all mean values for $L < 47$ we report are actually on a given regression line. For example, for the the ES-solver, we observe data points up to a factor of 2.0 above and 1.7 below the nominal *ES_samples predictor* – until we reach the value of $L = 43$. At this value, we record a nearly 14-fold increase above the nominal *ES_samples predictor*. Also, there is a reduction by a factor of 4.3 at $L = 46$. Such variability is statistically significant as it clearly exceeds the [3/4, 4/3] bounds expected due to statistical variability alone. As our detailed analysis in the next section suggests, the variability of the solver may be due to two factors: (1) significant change of LABS problem landscape with change of L since the number of optimal solutions may vary significantly as illustrated in Table 1, and (2) partial mismatch of ES-solver parameters for the given value of L. Additional consecutive data points are needed to more completely characterize the asymptotic performance of the ES-solver; its performance variability may or may not exceed the variability demonstrated for the KL-solver up to the value of $L = 64$ in the section that follows.

For the KL-solver, we observe data points up to a factor of 4.0 above and 3.6 below the nominal *KL_samples predictor* – uniformly across the observed range. Again, such variations by far exceed the expected statistical variations of the mean values. However, as shown in Figure 5, the sampling efficiency for the KL-solver is subject to noticeably smaller variations than the sampling efficiency for the ES-solver. We thus conjecture that the variability observed for the KL-solver is mostly due to the variability of the LABS problem landscape with change of L.

[2]We remind the reader that when we observe that distribution is exponential, *mean*≈standard deviation

[3]Only the *consecutive* values of $L = 20$ to $L = 47$ are used in the least-square fit to each data set. Data points for the ES-solver for $L = 50, 54, 58$ are shown for illustration only.

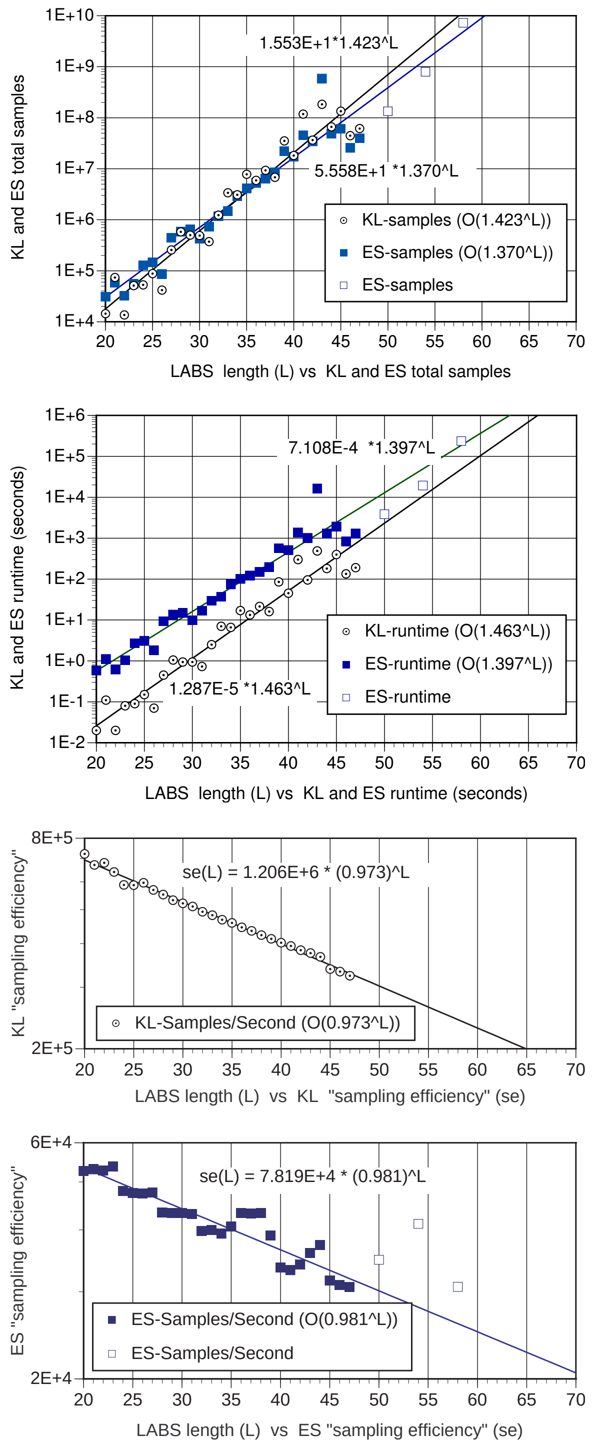


Figure 5: Asymptotic comparisons of ES and KL solvers.

Asymptotically, the nominal performance of the ES-solver appears better: there is a crossover of the *samples* predictors at $L = 34$ and the *runtime* predictors at $L = 88$. However, these crossovers are not well-defined due to the demonstrated variability and most importantly, there is a significant difference in the *sampling efficiency* between the two solvers; for $L = 34$ the average *runtimes* are 6.60 and 74.5 seconds for KL and ES, respectively. A significant improvement of the sampling efficiency of the ES solver is needed to make it competitive, for $L < 88$, with the current version of the KL solver.

Experiments as report here were executed on 266 MHz workstation under Linux. The current platform is clearly unsuitable for experiments at $L > 88$ even with the faster-of-the-two solvers. For $L = 77$, the *projected runtime* (using 1.552e-8*1.419^L for the KL-solver) is 0.94 years; if the processor performance would increase tenfold, it would still take 1.1 years to complete similar runs for $L = 84$. What is needed is a LABS solver with *both* (1) runtime complexity significantly better than $O(1.419^L)$ *and* (2) constant factor significantly better than 1.552e-8 relative to our 266 MHz platform.

However, as presented in the next section, we can demonstrate the reliability of the cost predictor under the termination 'B' for the KL-solver in the range of $48 \leq L \leq 64$ and slightly beyond.

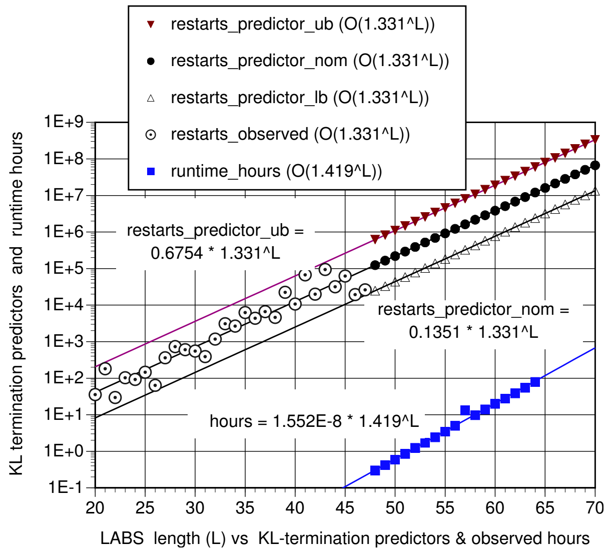
5 Cost Predictions under Termination 'B'

We ask the question: 'How long should the KL solver run in order to find the optimum for a given L with a certain probability of success P_{succ}?' We find the answer by analyzing the asymptotic behavior of either *runtime*, *samples* or *restarts* of the KL solver under termination criterion 'A'. An example of *restarts* data is shown in the top part of Figure 6. The average values of restarts are based on 128 experiments for consecutive values of $L = 20$ to $L = 47$. Also shown are three regression lines derived from the observed data for $L \leq 47$ and data points as predicted by the respective regression lines for values of $L > 47$:

$$\begin{aligned} \textit{nominal restarts predictor} &= 0.1351 \times 1.331^L & (9)\\ \textit{upper bound restarts predictor} &= 0.6754 \times 1.331^L & (10)\\ \textit{lower bound restarts predictor} &= 0.0270 \times 1.331^L & (11) \end{aligned}$$

According to the *nominal restarts predictor* in (9), the number of *required restarts* to find the optimum with $P_{succ} = 0.632$ for $L = 48, 60$, and 64, is 123,851 or 3,832,423 or 12,031,820, respectively. However, not all average restarts reported by the KL-solver are 'on the line' that corresponds to the *nominal restarts predictor*. In fact, the *maximum deviations* from the nominal predictions for *required restarts* for the range of $20 \leq L \leq 47$ are very significant: for $L = 26$ the prediction is far too conservative (229 restarts predicted, 64 observed, a ratio of 3.59), for $L = 41$ the prediction is far too optimistic (67,375 restarts observed, 16,726 predicted, a ratio of 4.03). Such variations by far exceed the expected statistical variability of the observed mean values of restarts: the 95% confidence intervals for the observed mean values are [52 ... 77] for $L = 26$ and [55,527 ... 79,222] for $L = 41$, respectively. The empirical *upper bound restarts predictor* in equation (10) simply multiplies the *nominal restarts predictor* by a factor of 5 which is based both on the worst-case ratio of 4.03 and the observed confidence interval for mean restarts at $L = 41$. Using the *upper bound restarts predictor*, the number of *required restarts* for $L = 41$ has the value of 81,380. Similarly, the empirical *lower bound restarts predictor* divides the *nominal restarts predictor* by a factor of 5. Clearly, such bounds may need to be adjusted for larger values of L when more experimental data becomes available.

(a) restarts predictions for the termination criterion 'B'



(b) summary of experiments based on predictions above

L	E_{best}^{1}	E_{min}	nExpr	nHits	uSols	hitR	hours[2]
48	140[6]	140	16	11	7	0.6875	0.3
49	136[6]	136	16	12	9	0.75	0.4
50	153[6]	153	16	15	11	0.9375	0.6
51	153[6]	153	16	5	3	0.3125	0.8
52	166[6]	166	16	12	7	0.75	1.2
53	170[6]	170	16	12	8	0.75	1.7
54	175[6]	175	16	10	6	0.625	2.4
55	171[6]	171	16	14	7	0.875	3.5
56	192[6]	192	16	16	7	1	5.0
57	188[6]	188	16	11	3	0.6875	13.2
58	197[6]	197	16	13	7	0.8125	9.8
59	205[6]	205	16	16	9	1	13.9
60	218[6]	218	16	16	10	1	20.0
61	226[12]	226	16	10	6	0.625	27.7
62	235[12]	235	16	16	10	1	38.8
63	207[12]	207	8	6	2	0.75	55.0
64	208[12]	208	4	2	2	0.5	78.6

[1]values shown for $L < 61$ are known to be optimal.
[2]runtime required to execute each experiment on a 266 MHz workstation under Linux.

Figure 6: Termination criterion 'B' results with KL solver.

The effectiveness of these predictors as the termination criterion 'B' is demonstrated by a series of experiments with $48 \leq L \leq 64$ on a cluster of 4 workstations, summarized in the table in Figure 6. Using the *nominal restarts predictor* in equation (9), we performed 16 experiments for each $48 \leq L \leq 62$, 8 experiments for $L = 63$, and 4 experiments for $L = 64$. The columns reported in the table refer to sequence length (L), best known minimum (E_{best}), best minimum reported by KL solver (E_{min}), number of experiments (*nExpr*), number of times E_{min} is found (*nHits*), number of unique solutions reporting E_{min} (*uSols*), hit ratio of *nHits* versus *nExpr* (*hitR*), runtime required to execute each experiment (*hours* on a 266 MHz workstation under Linux).

Assuming that the *nominal restarts predictor* values that terminate each run are 'correct', we expect each experiment to find the optimum with $P_{succ} = 0.632$, i.e. we expect all E_{min} to match or improve on E_{best} and the hit ratio to be 0.632. For 17 values of L in this exploratory study, we observe a hit ratio mean of 0.768 with a standard deviation of 0.191 – somewhat better than expected. The most likely reason is that for most values of L in this range, the true (and unknown) mean restart values are below the regression line in Figure 6. For example, finding 16 hits out of 16 experiments for $L = 60$ implies that the predicted mean value of 3,832,423 restarts should really be *reduced* by a factor of 3, 4, or 5.

Now, we cannot say that the values of E_{min} we report for for $L > 60$ are exact optima. However, given the experimental results at hand, the minima reported for 16 experiments with a hit ratio greater than or equal to 0.625 are the likely optima. If this threshold is not maintained, the number of restarts must be increased until we reach the threshold. To increase the overall reliability of the predictions, we can also increase the number of experiments while maintaining the hit ratio close to 0.632. Increasing the number of experiments for $L = 63, 64$ is in progress, with experiments for $L > 64$ to follow.

Compared to performance of earlier algorithms, the performance of the KL solver under the termination criterion B as shown in the graph and the last column of the table in Figure 6 is very effective: for $L = 60$ it took $16 \times 20 = 320$ hours to find not only one optimum solution but 16 solutions from which 10 optima were found to be unique. As shown in the graph, the *runtime* complexity of this solver is $O(1.42^L)$. The question arises, how much better can we do with an alternative solver.

6 Updates on Current Experiments

Since the two LABS solvers in this work are so different, a number of interesting question arose during and after the presentation of the preliminary work on this project [15]:

- Why does the KL-solver always restart from a strictly random bit string?
- What happens if the ES solver generation is periodically restarted with random parent strings?
- How 'optimal' are the settings for the ES-solver?

We conducted additional experiments to provide some in insights to these questions.

Settings for ES-solver. The parameters used for all ES-solver experiments reported in earlier sections are based on settings evaluated as 'best overall' in [10]: $\lambda = 10$ and $\mu = 3 * L$. We conducted a number of experiments under termination 'A' where we vary the size of ES

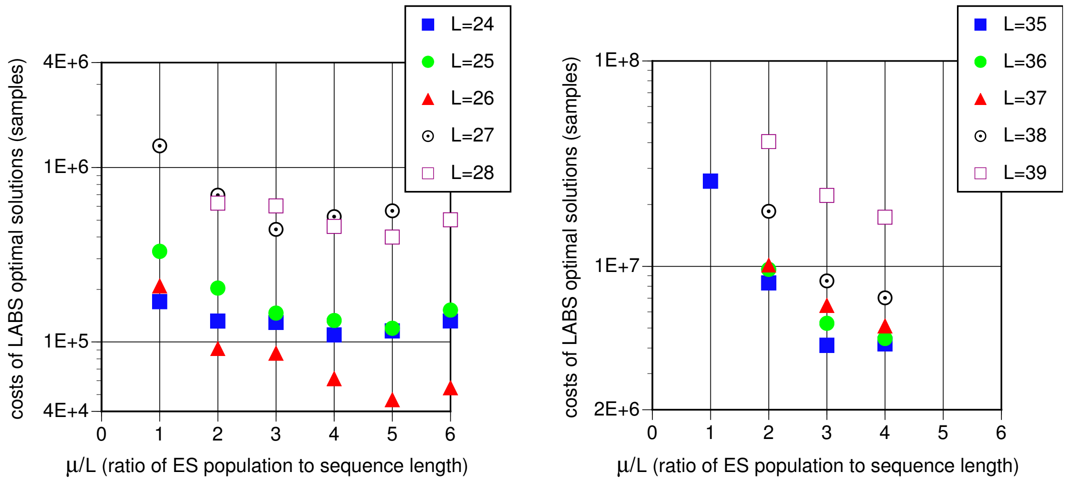


Figure 7: The effect of varying the size of ES population μ on the cost of finding the optimal solution. The cost is reported as the average total number of samples (based on 128 experiments).

population such that the μ/L ratio varies in increments of 1, from 1 to 6 typically. Results of 128 experiments for values of L =24–28 and L =35–39 are shown in Figure 7: rather than reporting *runtime* as the average cost, we report the number of *samples* required to find the specified optimum energy. Notably, the ratio of μ/L is 3 only for $L = 27$ and $L = 35$. For other values, the better ratio may be 4, 5, and possibly 6. This sensitivity to the value of this ratio as L is varied may be a factor in the variability we observe in the earlier experiments with the ES-solver.

Restarts for ES-solver? In these experiments, we keep the nominal values of $\lambda = 10$ and $\mu = 3 * L$ constant. However, we introduce the ratio $gM = generations_{max}/L$. We now launch 128 experiments under the termination 'A' for several values of gM and observe the number of average number of samples required to find the optimum for each setting. When $gM \rightarrow \infty$, we have the nominal ES-solver with 0 restarts. For $gM < \infty$, several restarts may be required before the ES-solver finds the optimum value and terminates. We show the result of experiments for $L = 27, 28$ in Figure 8. Note that the variability ratio in the average number of samples relative to the value reported for 0-restarts is within an interval [3/4, 4/3] which implies that for $L = 27, 28$, we cannot distinguish between the ES-solvers with no restarts and with restarts – as long as the ratio $gM > 1$.

Improved restarts for KL-solver? There are problems and solvers for which restarts that 'remember' some fraction of earlier local optimum solution may benefit the average solver performance, e.g. [25]. However, as our experiments in Figure 9 demonstrate, this heuristic does not seem to work well with the LABS problem and the KL solver. In fact, we don't know how much we need to perturb the currently returned local optimum solution before we restart. The experiments for $L = 27, 28$ demonstrate that the KL-solver has the best performance when each restart takes place from an completely random string.

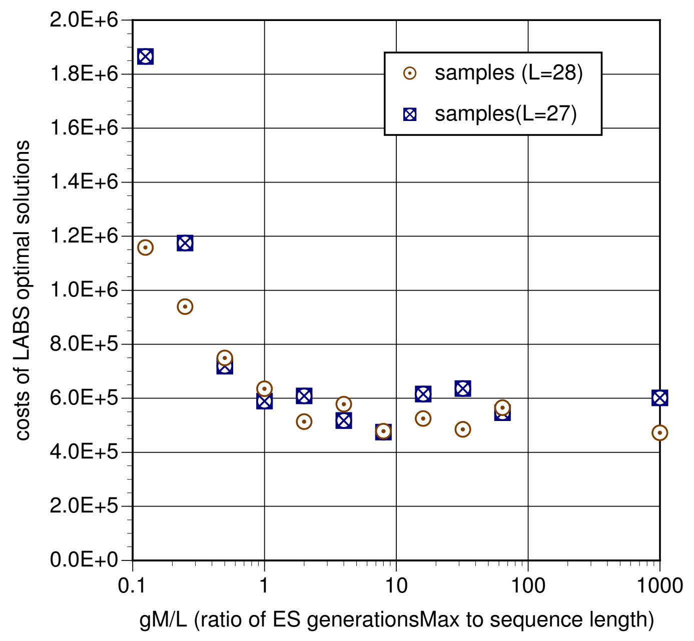


Figure 8: The effect of varying the *maximum number of ES generations before a random restart* on the cost of finding the optimal solution. Upon reaching the specified maximum, a new generation restarts from a randomly selected seed. No restarts are required for the right-most point (at $gM/L = 1000$). The cost is reported as the average total number of samples (based on 128 experiments).

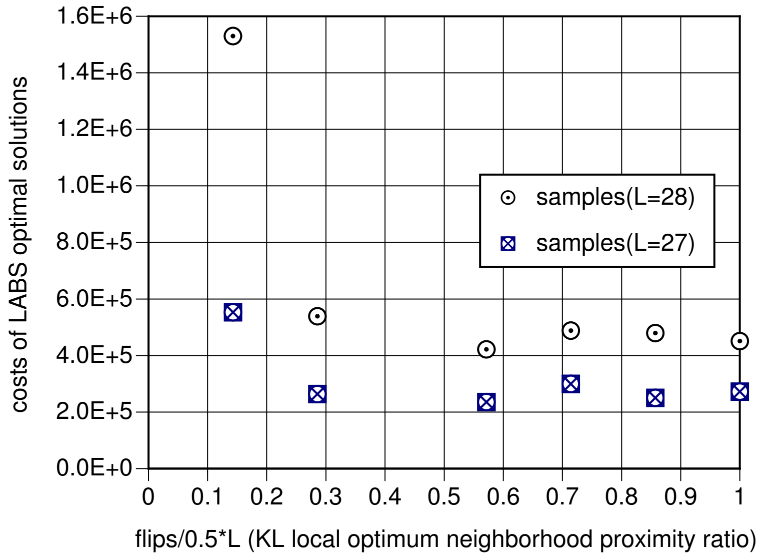


Figure 9: The effect of varying the distance of previous best solution to initialize each successive KL-restart. We vary the distance by randomly flipping k bits, $k \leq L$, in the best local solution. The *KL local neighborhood proximity ratio* is defined as $2k/L$. The cost is reported as the average total number of samples (based on 128 experiments). Results reported for ratio of 1 are statistically equivalent to experiments with strictly random initialization of each KL-restart.

7 Conclusions and Future Work

We have demonstrated a methodology that, while not guaranteeing optimum solutions for LABS problem, finds optima with high degree of confidence. For values of $L < 88$, the present solver of choice is KL since it has similar scaling properties to ES while generating new sequences at a significantly faster rate. Our methodology, to compare two LABS solvers not only in terms of *runtime* but also *samples* required to find optimal solutions, provides new insights about the effectiveness and the efficiency of evolutionary strategies. Compared to KL, the ES-based search for optima may require significantly less samples (on average) as the problem size increases, however the challenge to reduce the cost of computing these samples should be addressed next by the ES-based search algorithms. The challenge for KL-based algorithms is clearly to reduce the number of samples required to find optimal solutions.

A unique feature, intrinsic to the approach in this paper is the generation of multiple solutions for a given value of L. The number of these solutions in Table 1, introduced earlier is exact for values of $L \leq 35$. For values of $L > 35$, the values listed are conjectured from our experiments to date. A total of 400 experiments under termination 'A' have been run for $L <= 47$, so values listed are most likely exact. Values for $L > 47$ are based on smaller number of experiments as listed in the table of Figure 6. Since it is clear that the number of solutions must always be a multiple of 4, we have rounded the solutions found to the nearest such value. For example, when we report the number of solutions for $L = 60$ as 10 in Figure 6, we report 12 in the table above. We anticipate new insights by (a) correlating the number of solutions to the observed and predicted variability in solver performance and (b) analyzing solution structures to improve the search process for the next generation of LABS solvers.

We have just gained access to a cluster of 128 processors and plan to continue with scaling up the experiments with improved solvers, not only to search for new optima but also to find solutions for $L > 100$ that wil exhibit a converging trend towards the posted asymptotic value. Reader is invited to visit the *LABSbed* home page [24], not only to access complete archives of experimental data summarized in this paper but also to access updates on (1) on LABS minima for values of $L > 64$ and a call for collaboration, (2) improved performance of new LABS solvers, (3) status of LABSbed tutorial, documentation, and ready-to-install solvers.

Acknowledgements. The ready access to the exact minima of the LABS problem posted by Stephan Mertens [6] has been of great value in this research. The productive participation of the undergraduate cluster computing project team led by Jonathan B. Cage was instrumental for the LABS validation experiments for $L = 48$–60 and the LABS frontier experiments for $L = 61$–64. The just-in-time table of best merit factors [12] sent to us by Joshua Knauer also provided an update of our Figures 1 and 6.

References

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

[2] J. Bernasconi. Low autocorrelation binary sequences: statistical mechanics and configuration space analysis. *J. Phys.*, 48:559–567, April 1987.

[3] Martin Pelikan and David E. Goldberg. Hierarchical boa solves Ising spin glasses and MAXSAT. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 1271–1282, 2003.

[4] Martin Pelikan, David E. Goldberg, Jiri Ocenasek, and Simon Trebst. Robust and scalable black-box optimization, hierarchy, and ising spin glasses. IlliGAL Report No. 2003019, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 2003.

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

[6] S. Mertens. Ground states of the bernasconi model with open boundary conditions, 2003. For details, see http://odysseus.nat.uni-magdeburg.de/~mertens/bernasconi/open.dat.

[7] M. P. Eastwood, C. Hardin, Z. Luthey-Schulten, and P. G. Wolynes. Evaluating protein structure-prediction schemes using energy landscape theory. *IBM Journal of Research and Development*, 45(3/4), 2001. Special issue on *Deep computing for the life sciences*.

[8] C. de Groot, D. Wurtz, and K. H. Hoffmann. Low autocorrelation binary sequences: Exact enumeration and optimization by evolutionary strategies. *Optimization (UK)*, 23(4):369–384, 1993.

[9] F.-M. Dittes. Optimization on rugged landscape: A new general purpose monte carlo approach. *Physical Review Letters*, 76:4651–4655, 1996.

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

[11] S. Prestwich. A Hybrid Search Architecture Applied to Hard Random 3-SAT and Low-Autocorrelation Binary Sequences. *The Sixth International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, Springer-Verlag*, 1894:337–352, 2000.

[12] J. Knauer. Home Page of LABS Problem Merit Factor Records, 2003. See http://www.cecm.sfu.ca/~jknauer/labs/records.html.

[13] F. Brglez, X. Y. Li, and M. Stallmann. On SAT Instance Classes and a Method for Reliable Performance Experiments with SAT Solvers. *Annals of Mathematics and Artificial Intelligence (AMAI),* , 2003. To be published. A major revision of the 2002-SAT Symposium paper. For a preprint, see `http://www.cbl.ncsu.edu/publications/` .

[14] F. Brglez, M. F. Stallmann, and X. Y. Li. The LABS Problem and an Encapsulation of Local Search Algorithms for Improved Performance and Reliability. *Technical Report*, 2003-TR@CBL-LABS, 2003. For a reprint, see http://www.cbl.ncsu.edu/publications/.

[15] F. Brglez, M. Stallmann, X. Y. Li, and B. Militzer. Reliable Cost Predictions for Finding Optimal Solutions to LABS Problem: Evolutionary and Alternative Algorithms. In *Proceedings of The Fifth International Workshop on Frontiers in Evolutionary Algorithms (FEA2003), Cary, NC, USA, September 26-30, 2003* , September 2003. See also `http://scsx01.sc.ehu.es/-ccwgrrom/FEA2003/` and `http://www.cbl.ncsu.edu/OpenExperiments/LABS/` .

[16] I. Rechenberg. Evolutionsstrategien - optimierung technischer systeme nach prinzipien der biologischen information. *Stuttgart-Bad Cannstatt: Friedrich Frommann Verlag*, 1973.

[17] H.-P. Schwefel. *Numerical Optimization of Computer Models*. New York: Wiley, 1981.

[18] B.W.Kernighan and S.Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, pages 291–307, 1970.

[19] Woei-Kae Chen and Matthias F.M. Stallmann. Local search variants for hypercube embedding. In *Proceedings Fifth Distributed Memory Computing Conference*, pages 1375 – 1383, 1990.

[20] W.-K. Chen, M. Stallmann, and E.F. Gehringer. Hypercube embedding heuristics: An evaluation. *International Journal on Parallel Programming*, 18(6):505 – 549, 1989.

[21] Holger H. Hoos and Thomas Stützle. Local Search Algorithms for SAT: An Empirical Evaluation. *Journal Of Automated Reasoning*, 24, 2000.

[22] D. Johnson. A Theoretician's Guide to the Experimental Analysis of Algorithms. *American Mathematical Society*, 220(5-6):215–250, 2002. In Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges, M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, Editors.

[23] F. Brglez, M. F. Stallmann, and X. Y. Li. SATbed: An Environment For Reliable Performance Experiments with SAT Instance Classes and Algorithms. In *Proceedings of SAT 2003, Sixth International Symposium on the Theory and Applications of Satisfiability Testing, May 5-8 2003, S. Margherita Ligure - Portofino, Italy*, 2003. For a reprint, see `http://www.cbl.ncsu.edu/-publications/` .

[24] LABSbed: LABS Problem Experiments Home Page, 2003. See http://www.cbl.ncsu.edu/-OpenExperiments/LABS/.

[25] K. Smyth, H. Hoos, and T. Stutzle. Iterated robust tabu search for MAX-SAT. In *Proceedings of the 16th Canadian Conference on Artificial Intelligence (AI 2003)*, 2003.

Appendix: Generalization of the Kernighan-Lin Heuristic

The KL (Kernighan-Lin) heuristic, introduced in the context of the LABS problem, has been illustrated as a simplified flowchart in Figure 3. This appendix provides additional background and concludes with a pseudo code description of the generalized Kernighan-Lin algorithm [19, 20].

The key ingredients to the *generic KL* algorithm are as follows:

1. The concept of a *feasible solution*, which encompasses any combinatorial object that is encountered during the search. In graph partitioning, any partition of the vertices is considered a feasible solution (originally, Kernighan and Lin insisted on a partition that divided the vertices equally [18]). In the LABS problem, any binary sequence of the specified length is a feasible solution.
2. A *cost function* c that maps any feasible solution x to its cost $c(x)$. We will assume, without loss of generality, that $c(x)$ is a natural number and that an optimum solution $\hat{x}$ minimizes $c(\hat{x})$. The cost function is not necessarily the one defined by the problem — it may be designed to guide the search toward solutions from which an optimum is easier to find.
3. A set possible transformations that define the *neighborhood* $\mathcal{N}(x)$ of a feasible solution x. Every element of $\mathcal{N}(x)$ must be a feasible solution and must be derived from x via one of the specified transformations. The original Kernighan-Lin heuristic considered any transformation that swapped two vertices, one from each side of the partition. Later variations looked at movements of a single vertex from one side to the other (with a penalty for unbalanced partitions built into the cost function). In the LABS problem, a transformation is the flip of a single bit and $\mathcal{N}(x)$ is the set of all strings that are Hamming distance 1 from x.
4. A *move* occurs when the algorithm chooses a neighbor of the current feasible solution x and makes it the new feasible solution.
5. A *marking* strategy that guarantees that no move will be reversed during a single *sweep*. In the original Kernighan-Lin heuristic a vertex was marked whenever it was used in a swap, a marked vertex could not be used again in the same sweep, and the end of a sweep occurred when all vertices were marked (at which point the next sweep began by unmarking the vertices). In the LABS problem a bit position is marked when that bit is flipped and no other flip can occur at that bit position during a sweep. The marking strategy must satisfy two properties: (a) it must prevent the reversal of a move performed during a sweep, and (b) it must terminate, that is, every move must make progress toward a situation where the marks prevent any further move (and the end of a sweep occurs).

The algorithm is shown in Figure 10.

Generic KL Algorithm

```
$bestCost \leftarrow \infty$
for the desired number of restarts do
    Pick a random feasible solution $x$
    $bestRepCost \leftarrow c(x)$
    $bestRepSolution \leftarrow x$
    $improvement \leftarrow$ TRUE
    while $improvement$ do
        ▷ perform another sweep
        $improvement \leftarrow$ FALSE
        Erase all marks
        $bestSweepCost \leftarrow c(x)$
        $bestSweepSolution \leftarrow x$
        while an unmarked move is possible do
            Choose $y \in \mathcal{N}(x)$ so that $c(y)$ is minimized
            Mark the move from $x$ to $y$
            $x \leftarrow y$
            if $c(x) < bestSweepCost$ then
                $bestSweepCost \leftarrow c(x)$
                $bestSweepSolution \leftarrow x$
            endif
        end do    ▷ while move is possible
        $x \leftarrow bestSweepSolution$
        if $c(x) < bestRepCost$ then
            $bestRepCost \leftarrow c(x)$
            $bestRepSolution \leftarrow x$
            $improvement \leftarrow$ TRUE
        endif
    end do    ▷ while improvement is possible
    if $bestRepCost < bestCost$ then
        $bestRepCost \leftarrow bestCost$
        $bestRepSolution \leftarrow bestRepSolution$
    endif
end do    ▷ for ... restarts
```

Figure 10: The generalized Kernighan-Lin algorithm.