

A Hybrid Search Architecture Applied to Hard Random 3-SAT and Low-Autocorrelation Binary Sequences

Steven Prestwich

Department of Computer Science
National University of Ireland at Cork
`s.prestwich@cs.ucc.ie`

Abstract. The hybridisation of systematic and stochastic search is an active research area with potential benefits for real-world combinatorial problems. This paper shows that randomising the backtracking component of a systematic backtracker can improve its scalability to equal that of stochastic local search. The hybrid may be viewed as stochastic local search in a constrained space, cleanly combining local search with constraint programming techniques. The approach is applied to two very different problems. Firstly a hybrid of local search and constraint propagation is applied to hard random 3-SAT problems, and is the first constructive search algorithm to solve very large instances. Secondly a hybrid of local search and branch-and-bound is applied to low-autocorrelation binary sequences (a notoriously difficult communications engineering problem), and is the first stochastic search algorithm to find optimal solutions. These results show that the approach is a promising one for both constraint satisfaction and optimisation problems.

1 Introduction

The *low-autocorrelation binary sequence* (LABS) problem has many applications to communications and electrical engineering, and a long history. The problem is as follows. Consider a binary sequence $S = (s_1, \dots, s_N)$ where each $s_i \in \{1, -1\}$. The *off-peak autocorrelations* of S are defined as:

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

and the *energy* of S is defined as:

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

The LABS problem is to assign values to the s_i such that $E(S)$ is minimised. A standard measure of sequence quality is the *merit factor* $F = N^2/2E$. Theoretical considerations [13] give an upper bound of $F \approx 12.32$ for $N \gg 1$, and

empirical curve fitting on known optimal sequences [23] yields an estimate of $F \approx 9.3$. LABS is of interest to the Constraints community because of its difficulty; it is problem number 5 in the CSPLIB benchmark library [8], a web-based collection of constraint problems. Since the 1970s researchers, many from the Physics community, have applied various methods to the problem. Analytical methods have been used to construct optimal sequences for certain values of N using *periodic boundary conditions* (see [24] for example), but for the general *aperiodic* case search is necessary. Two possibilities are *systematic* and *stochastic* search. LABS is particularly challenging because it poses difficulties for both search paradigms.

Systematic search often involves the enumeration of possibilities by backtracking, using whatever constraint programming techniques are available to eliminate redundant parts of the search space. Golay [13] used exhaustive enumeration to find optimal sequences for $N \leq 32$. Mertens [23] enumerated optimal sequences for $N \leq 48$ using systematic search augmented with two techniques to reduce the size of the search space: *branch-and-bound* and *symmetry breaking*. Branch-and-bound improves performance significantly: a systematic search for sequences of length 44 took approximately 2 days, as opposed to an extrapolated 68 days for exhaustive enumeration.¹ Symmetry breaking exploits the fact that sequences occur in equivalence classes of size 8. Gent & Smith [10] have also successfully applied *generic* symmetry breaking techniques to LABS. However, even with these enhancements, systematic search is unlikely to scale up to large sequences. LABS provides little scope for constraint propagation, and it is conjectured [23] that for $N > 100$ progress will be made through mathematical insight rather than computer power.

When systematic search breaks down, a standard approach for solving combinatorial problems is to resort to stochastic search: simulated annealing, evolutionary algorithms, neural networks, ant colonies or hill climbers. Such algorithms are often able to solve much larger instances. Unfortunately, they perform quite poorly on some problems, and finding optimal LABS solutions seems to be an example. This is despite decades of research effort using simulated annealing [3,13], evolutionary algorithms [6], analogies with molecular evolution [38] and several other heuristic search algorithms [2]. The cause is considered to be the search space normally used for LABS (the space of all sequences of a given length) whose cost function E has a very irregular structure with isolated minima [3].

A technique that may be used with both systematic and stochastic search is *sieves* to restrict the search to a promising subspace. The most common sieve is *skew-symmetry*: only sequences of length $N = 2n - 1$ satisfying $s_{n+i} = (-1)^i s_{n-i}$ ($i = 1 \dots n - 1$) are considered, roughly halving the number of independent variables in the problem and greatly reducing the search space size. Skew-symmetric sequences have been enumerated for $N \leq 71$ [6] and often have rather good merit factors because $C_k = 0$ for all odd k . However, stochastic search finds large skew-symmetric sequences with only $F \approx 6$.

¹ On a Sun UltraSparc I 170 workstation.

Thus neither systematic nor stochastic search seems suitable for finding large optimal sequences. This situation is mirrored in some other combinatorial problems, and though both systematic and stochastic search are highly successful, neither is seen as adequate for all problems. Hence a recent trend in Artificial Intelligence toward *hybrid* search, with the aim of combining advantages of both. A variety of approaches have been described (see Section 5) but most allow constraint violations. The hybrid approach described in this paper is inspired by the observation that systematic search is able to exploit constraint programming techniques but sometimes scales poorly; perhaps by sacrificing systematicity, scalability might be improved. Because a randomised backtracker can be viewed as a stochastic local search algorithm, we call this hybrid approach Constrained Local Search (CLS).

This paper is an expanded version of a technical report [28]. Previous papers on CLS have dealt with graph colouring [29] and satisfiability [30]. The paper is organised as follows. Section 2 briefly describes the CLS approach to constraint problems. Section 3 cites previous work on CLS for SAT problems, and gives new scaling results for hard random 3-SAT. Section 4 applies CLS to LABS. Section 5 discusses related work. Finally, Section 6 draws conclusions.

2 Constrained Local Search

CLS hybrid algorithms are constructed by randomising the backtracking component of a systematic algorithm; that is, allowing backtracking to occur on *arbitrarily-chosen* variables. The CLS schema is shown in Figure 1. A is the current set of assigned variables, initialised to $\{\}$. U is the current set of unassigned variables, initialised to the full set of variables $\{v_1, \dots, v_N\}$. $B \geq 1$ is an integer parameter called the *noise level*. The condition “assigning x to u is consistent” means that assigning the value x to the variable u does not violate any constraints after propagation, if constraint propagation is used. The algorithm proceeds by selecting unassigned variables using a selection rule VAR, and assigns values to them using a selection rule VAL. On reaching a dead-end ($D = \{\}$) it backtracks on B variables selected by a rule BACK, which selects variables either randomly or heuristically; in either case, no attempt is made to maintain completeness. Termination is not guaranteed but occurs if all variables are assigned ($U = \{\}$). The constraint propagation, the variable domains, the value of B , and the VAL, VAR and BACK rules are problem-dependent. The *unpropagation* of constraints is the hardest part of implementing a new CLS algorithm: the effects of an arbitrary variable on the constraint store must be undone. For details on the graph colouring and SAT implementations see [29,30] respectively. Unpropagation is easy for LABS, where the only constraint reasoning used is branch-and-bound (see Section 4).

Why should randomised backtracking be expected to perform well? Intuitively, it appears inferior to intelligent backtracking schemes, and it has the disadvantage that completeness is lost. The rationale is that randomised backtracking can be viewed as a *stochastic local search* algorithm. Local search ex-

```

function CLS( $B$ ):
   $A = \{\}$ 
   $U = \{v_1, \dots, v_N\}$ 
  while  $U \neq \{\}$ 
     $u = \text{VAR}(U)$ 
     $D = \{x \in \text{domain}(u) \mid \text{assigning } x \text{ to } u \text{ is consistent}\}$ 
    if  $D = \{\}$ 
      for  $i=1$  to  $\min(B, |A|)$ 
         $a = \text{BACK}(A)$ 
        unassign  $a$  and unpropagate
         $A = A - \{a\}$ 
         $U = U \cup \{a\}$ 
      else
        assign  $\text{VAL}(D)$  to  $u$  and propagate
         $A = A \cup \{u\}$ 
         $U = U - \{u\}$ 
  return current assignments

```

Fig. 1. The CLS randomised backtracking schema

plores the *neighbourhood* of a point σ in a space by making *local moves*. The neighbourhood consists of the set of points σ' that can be reached by a single local move. The aim is to minimise (or equivalently to maximise) some *objective function* $f(\sigma)$ on the space. A local move $\sigma \rightarrow \sigma'$ can be classified as *backward*, *forward* or *sideways*, depending on whether $f(\sigma') - f(\sigma)$ is positive, negative or zero. Local search may converge on a *local minimum*: a point that has lower value than all its neighbours but is not a global minimum. The aim of backward moves is to escape from local minima by providing *noise*, while sideways moves are used to traverse function plateaus.

Local search is usually applied to optimisation problems (such as LABS) by defining the search space as the total assignments (LABS sequences of a given length), a local move as a change in the value of a single variable, and using the given objective function (the LABS sequence energy). It is often applied to constraint satisfaction problems (such as SAT) by searching the same space with the same local moves, but using a measure of constraint violation as the objective function. CLS can be viewed as a local search algorithm whose search space is the consistent partial assignments; variable assignments are forward local moves, backtracks are backward local moves, and the number of unassigned variables is the objective function to be minimised. The crucial difference between CLS and most other local search algorithms is that, like systematic backtracking, it is *constructive*: it violates no constraints and can therefore exploit techniques such as forward checking (for SAT [30]), branch-and-bound (for LABS) or both (for graph colouring [29]).

It might be objected that *all* backtracking algorithms are local search algorithms in the same sense as CLS: they search the same space and try to minimise

the same function. However, systematic backtracking has so far failed to achieve the scalability of local search. There seems to be a missing ingredient, though there is currently no consensus on what the ingredient is. An obvious candidate is randomness, but research has shown that randomness is not a necessary property for scalability [9,25]. Likely candidates are hill-climbing and freedom of movement in the search space; see [7] for a debate on this issue. Whatever the ingredient is, we aim to show that CLS has it.

3 CLS for SAT

The SAT problem is to determine whether a Boolean expression has a satisfying set of truth assignments. The problems are usually expressed in conjunctive normal form: a conjunction of clauses $C_1 \wedge \dots \wedge C_m$ where each clause C is a disjunction of literals $l_1 \vee \dots \vee l_n$ and each literal l is either a Boolean variable v or its negation $\neg v$. A Boolean variable can be labeled *true* or *false*. The CLS implementation for the SAT problem described in [30] will be used, which maintains consistency by forward checking. The noise level B is set to 1 for all problems. CLS has previously been applied to several SAT benchmarks. Here we evaluate it on a standard benchmark: hard random 3-SAT. The random 3-SAT problem is the SAT problem where each clause contains three literals, each literal being negated with probability $\frac{1}{2}$. The *hard* problems are in the *crossover* region where 50% of problems are satisfiable, which occurs when the ratio of clauses c to variables v is approximately $c/v = 4.258 + 58.26v^{-5/3}$ [5].

A note on experimental details. Throughout this paper, CLS is executed on a 300 MHz DEC Alphaserpver 1000A 5/300 under Unix. Times are in seconds, and those from other papers are normalised to the Alphaserpver using benchmark timings from [18]. For readers wishing to normalise our execution times to other platforms, the DIMACS² benchmark program `dfmax r500.5` takes 46.2 seconds on the Alphaserpver.

First we consider published results by Parkes & Walser [26], who apply the WSAT local search algorithm to these problems. They use a statistical technique called *retrospective parameter variation* to optimise a WSAT parameter. Using between 10^3 and 10^4 runs they find the median number of *flips* (changes to the truth value of a variable) necessary to find solutions to problems of various sizes. Solvable problems were generated using a systematic SAT algorithm (TABLEAU [5]) then passed to WSAT. Results for CLS were generated differently: it was applied to 10^3 problems of each size, and the 25th percentile taken of the execution time to find a solution. Because 50% of problems from the crossover region are satisfiable, the 25th percentile for all problems is the median for the solvable problems. This is faster than using exhaustive search to generate solvable problems. Figure 2 shows median results for CLS and WSAT, counting WSAT flips and CLS backtracks required to find a solution. The results for CLS and WSAT are remarkably similar. Moreover, both curves show

² <ftp://dimacs.rutgers.edu/pub/challenge/>

a *less-than-exponential* increase, in contrast to chronological backtrackers. However, the relationship is not a simple monomial because a log-log plot shows a distinct upward curve. This property has also been noted by Gent & Walsh [9] for variants of GSAT (an ancestor of WSAT).

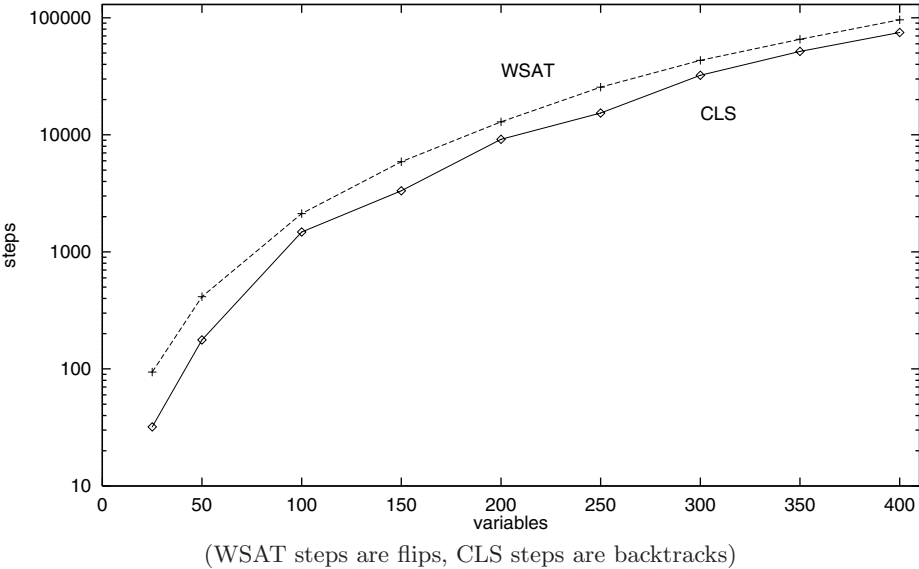


Fig. 2. Median steps on hard random 3-SAT problems

Next we consider published results by Selman, Kautz & Cohen [34] for GSAT with random walk, and by Shang & Wah [35] for DLM [35] (a recent local search algorithm based on Lagrange multipliers). Figure 3 compares mean times for the three algorithms on instances with 600, 1000 and 2000 variables. It is unclear whether these are precisely the same problem instances but the results are nevertheless comparable. The times for CLS are averaged over 10 runs. CLS is several times slower than both DLM and GSAT but it clearly has the scalability of a local search algorithm. In contrast to local search, systematic algorithms scale exponentially and cannot solve problems with more than approximately 400 variables [5]. An interesting experiment would be to compare the scalability of intelligent backtracking algorithms (such as RELSAT [1]). However, it has not been claimed that intelligent backtracking can solve such large 3-SAT instances, so we believe that such algorithms will also scale exponentially.

CLS backtracks are more expensive than WSAT, GSAT or DLM flips, as shown in the second experiment. Parkes and Walser found that WSAT performed approximately 70,000 down to 60,000 flips per second. CLS performed 31,716 down to 26,315 backtracks per second: a similarly small degradation but a slower rate, especially given the different platforms. CLS backtracks are also

problem	GSAT+w	DLM	CLS
f600	7.0	3.6	8.4
f1000	219	27.2	96.1
f2000	651	389	2145

Fig. 3. Mean times on hard random 3-SAT problems

more expensive than systematic backtracks, because of the book-keeping necessary to maintain consistency during randomised backtracking (though the cost of sophisticated variable ordering heuristics may outweigh this expense). In spite of this, on some large, structured SAT problems CLS out-performs both systematic and pure local search [30].

4 CLS for LABS

The starting point for the hybrid LABS algorithm is a simplified variant of Mertens' systematic algorithm, retaining branch-and-bound but not symmetry breaking (for reasons given below). Variables are initially *unassigned*: they have no assigned values. The simplified systematic algorithm proceeds by assigning either the value 1 or -1 to each variable in turn. A record E_{best} is maintained, which is the energy of the best solution found so far. E_{best} may be initialised to ∞ or some finite value above the known optimal energy, and on finding a better solution it is updated. Whenever an assignment causes the constraint $E < E_{best}$ to be violated (where E is the energy of the sequence under construction), backtracking occurs. Selecting a value for a variable s_i is the *branch* part of the algorithm, while the use of the constraint $E < E_{best}$ is the *bound* part. As the search proceeds E_{best} is refined, and more solutions are excluded from the search. Branch-and-bound's gain in efficiency comes from the fact that constraint violation can sometimes be detected before all variables have been assigned.

Constraint violation is detected as follows. Call a product $s_i s_j$ *computable* if s_i and s_j are both assigned. Then a lower bound E_L for the energy E is:

$$E_L = \sum_{k=1}^{N-1} L_k$$

where $L_k = \max(b_k, |T_k| - U_k)^2$, $b_k = (N - k) \bmod 2$, T_k is the sum of its computable products and U_k is the number of its non-computable products.³ Constraint violation occurs if $E_L \geq E_{best}$. The lower bound E_L is incrementally computed. On assigning a variable v the algorithm does the following for each occurrence $s_v s_w$ in some C_k where s_w is assigned: subtract L_k from E_L , decrement U_k , add $s_v s_w$ to T_k , recalculate L_k and add it to E_L . Conversely,

³ This is a slightly different lower bound to that used in [23] but is based on similar reasoning.

on unassigning a variable v , it does the following for each occurrence $s_v s_w$ in some C_k where s_w is assigned: subtract L_k from E_L , increment U_k , subtract $s_v s_w$ from T_k , recalculate L_k and add it to E_L . The occurrences of each variable v are stored in a table, together with the numbers w and k .

To obtain a hybrid LABS algorithm CLS is used in place of chronological backtracking, with the following heuristics:

- The noise level B is set to 1.
- The backtracking heuristic BACK selects random assigned variables.
- The dynamic variable ordering heuristic VAR usually selects the most recently unassigned variable, but if no improvement has occurred (in the number of assigned variables) after N backtracks then VAR selects a random unassigned variable. This produces a slowly-changing set of assigned variables.
- If $D = \{x\}$ then the value ordering rule VAL(D) returns x , otherwise it selects a value using a rule described below.
- A further heuristic is added: random restart. In case CLS becomes trapped in a deep local minimum, it is restarted at regular intervals (retaining the current E_{best} value). The interval used is 1.5^N backtracks.

Given a choice of values in D , VAL(D) selects as follows. It first tries to select a different value than the previous assignment for the given variable (previous assignments are stored in an array). On achieving this, future variables are assigned the same value as in their previous assignments, until backtracking next occurs. Then it tries to assign different values to variables again. The purpose of this heuristic is to reduce disruption to variable assignments as the set of assigned variables changes, while avoiding null local moves.

Figure 3 shows the results for the hybrid LABS algorithm: the sequence length N , the energy E , the merit factor F , the number of backtracks taken to find the sequence, the CPU time taken and an example of a sequence found by CLS. E_{best} was initialised to the known optimal energy plus 1 in each case. Mean times and backtracks were calculated over various numbers of runs: 10^6 ($N=3-6$), 10^5 ($N=7-10$), 10^4 ($N=11-15$), 10^3 ($N=16-17$), 10^2 ($N=18-20$), 10 ($N=21-41$) and 1 ($N=42-48$). Following convention, sequences are shown in *run-length notation*: each number indicates the number of consecutive elements with the same value. For example a sequence $(1, 1, -1, 1, -1, -1, -1, -1, 1)$ would be written 21141. For runs of length greater than 9, upper-case letters are used with A=10, B=11 etc. Note that 21141 could also represent the sequence $(-1, -1, 1, -1, 1, 1, 1, 1, -1)$. Sequences can have all their values inverted, or have alternate values inverted, or be reversed, without changing E .

The table shows that CLS is indeed able to find optimal sequences for $N \leq 48$, and it is the first stochastic search algorithm to do so. To estimate its scalability, execution times are plotted as a function of sequence length on a log-scale graph in Figure 5. Taking a least-squares fit, the estimated time taken to find an optimal solution is $O(1.698^N)$. Exhaustive search (without branch-and-bound) takes $O(2^N)$ and Mertens' algorithm takes $O(1.85^N)$ — that is to explore the *entire* search space, not simply to find the first optimal solution. However, the CSPLIB

N	E	F	backtracks	seconds	sequence
3	1	4.50	0	0.000008	12
4	2	4.00	0	0.000013	31
5	2	6.25	0	0.000026	311
6	7	2.57	0	0.000025	1131
7	3	5.44	5	0.00015	3211
8	8	4.00	4	0.00015	32111
9	12	3.38	4	0.00016	121113
10	13	3.85	4	0.00018	421111
11	5	12.10	15	0.0006	112133
12	10	7.20	40	0.0016	225111
13	6	14.08	75	0.0031	5221111
14	19	5.16	22	0.0011	42221111
15	15	7.50	360	0.0178	11213133
16	24	5.33	48	0.0042	11312143
17	32	4.52	11	0.0118	4221211112
18	25	6.48	1824	0.079	441112221
19	29	6.22	891	0.058	2122411114
20	26	7.69	9810	0.646	22124141111
21	26	8.48	52533	3.85	27221111121
22	39	6.21	8427	0.74	232212111115
23	47	5.63	28247	2.21	212121111632
24	36	8.0	114551	9.55	237111121221
25	36	8.68	10610	1.72	122121111733
26	45	7.51	36928	3.12	1122121111733
27	37	9.85	373197	33.9	34313131211211
28	50	7.84	407254	37.2	21211213131343
29	62	6.78	306246	43.3	212112131313431
30	59	7.63	53415	25.3	132311111212155
31	67	7.17	1135539	117	1112111122122337
32	64	8.00	3030725	319	12212233111121117
33	64	8.51	4322170	482	74211211111122221
34	65	8.89	3747743	422	84211211111122221
35	73	8.39	6100331	709	7122122111121111332
36	82	7.90	12207068	981	3632311131212111211
37	86	7.96	16596532	2096	122221111112112448
38	87	8.30	13033938	1679	2222942112112111111
39	99	7.68	40745979	5300	232411711111141122121
40	108	7.41	109581034	14305	111211211343143131312
41	108	7.78	38301376	21224	343111111222281211211
42	101	8.73	33446785	4890	211211211343143131313
43	109	8.48	324452495	46168	1132432111117212112213
44	122	7.93	166769694	27422	121112111222111311313525
45	118	8.58	367926334	52920	11111112343213212112128
46	131	8.08	35017866	5184	234232137111212212111111
47	135	8.18	164787014	26280	13211111111243212112A212
48	140	8.23	78278179	12096	24422321211321211111811

Fig. 4. Results on known optimal sequences

web site [8] contains a table of all optimal solutions for $N = 17 \dots 50$, and in 23 out of 34 cases there is only one optimal solution (modulo symmetry); nor does the number of optimal solutions increase exponentially or even monotonically with N . We therefore expect Mertens' algorithm also to take approximately $O(1.85^N)$ to find a solution, though this remains to be tested. Hence it seems that CLS scales better than the best systematic algorithm, as might be expected from a local search procedure.

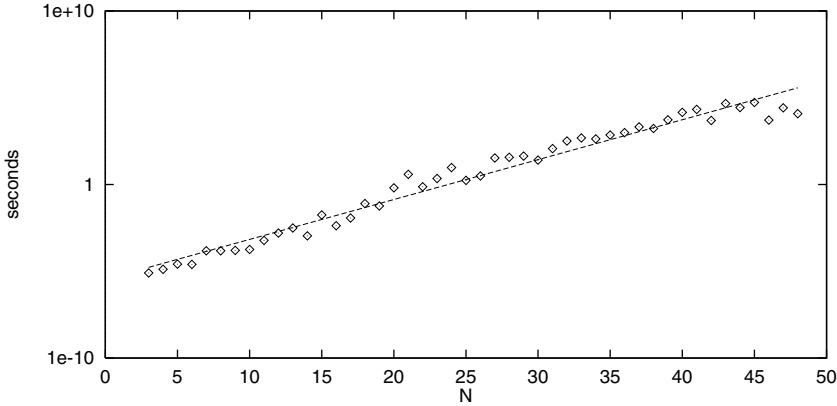


Fig. 5. Least-squares fit on known optimal sequences

The least-squares fit was made on the assumption that the curve has the form $a.b^N$ but this may not be the case. The graph has an interesting property: most of the points above the line occur in the middle of the graph, while those below the line occur mainly for low and high N . This *systematic variation* (to use statistics terminology) seems to indicate that execution time is a less-than-exponential function of N . If we estimate the scale factor based on $N \leq 25$ we obtain 1.887, for $N \leq 30$ we obtain exactly Mertens' factor of 1.850, and as we add the figures for larger N we obtain progressively smaller scale factors. As noted in Section 2, CLS and the local search algorithm WSAT exhibit similar sub-exponential scaling on hard random 3-SAT problems. The data is too noisy to draw a firm conclusion, but it would be remarkable if the same phenomenon occurs in LABS.

To investigate whether the algorithm is significantly slower when started from infinite energy, we repeated the experiment for $N \leq 30$. The least-squares fit to the formula $a.b^N$ gave a scale factor $b = 1.846$, which is very close to the factor for our original data for $N \leq 30$, and a coefficient a that is 2.29 times greater than before. In other words, the only effect of starting from infinite energy is to slow down the execution by a constant factor.

As a further experiment CLS was applied to sequences of length $N = 61$, which is currently too large for systematic search. de Groot et al [6] found

sequences up to $F = 6.5$ using evolutionary strategies. Bernasconi [3] found $F \approx 6.8$ using simulated annealing, though with worse average-case behaviour over many runs. The best result obtained by CLS after 2 days of computation was the sequence

1222122211111111B3236411211211

with $F = 7.1$ and $E = 262$. This is easily verified to be non-skew-symmetric and is the best such sequence known. However, better skew-symmetric sequences have been found [2,6], and in future skew-symmetry will be added to CLS.

A note on symmetry breaking. This technique was not used because it is likely to be of little use for stochastic search. The reasoning is that stochastic search explores only part of the search space, so it is the density and distribution of solutions that are important. Symmetry breaking reduces the size of the search space by some factor (up to 8 in the case of LABS) but it also reduces the number of solutions by a similar factor, leaving the density unchanged. Symmetry breaking may even have a harmful effect on stochastic search: if it applies extra constraints, and if these constraints cannot be invoked until several variables are assigned, then stochastic search may be lured into regions of the search space whose solutions are forbidden by the extra constraints.

5 Related Work

CLS has much in common with Ginsberg's Dynamic Backtracking (DB) algorithm [11], both being able to backtrack to earlier variables without disrupting those in between. However, DB's completeness requirement restricts its choice of backtracking variable. Ginsberg notes that this restriction cannot be lifted without losing completeness. CLS ignores the restriction and is therefore incomplete, but this pays off in scalability. Another related algorithm is Ginsberg & McAllester's Partial-order Dynamic Backtracking (PDB) algorithm [12]. PDB is a hybrid of the GSAT local search algorithm [34] and DB. It is complete and has more freedom of movement than DB, though to achieve complete freedom it requires exponential memory in the worst case.

Other non-systematic backtracking algorithms have been described. Langley's Iterative Sampling [22] constructs a random consistent partial assignment until reaching a dead-end, then restarts. This is essentially CLS with maximum noise (B set to the number of variables) and randomised heuristics. Harvey's Bounded Backtrack Search [16] is a hybrid of Iterative Sampling and chronological backtracking, alternating a limited amount of chronological backtracking with random restarts. Gomes, Selman and Kautz [15] periodically restart systematic SAT algorithms with slightly randomised heuristics, obtaining improved scalability on some problems. Goldberg & Rivenburgh's Restricted Backtracking [14] constructs maximum cliques using a trade-off between solution quality and search completeness. Joslin & Clements' Squeaky Wheel Optimization algorithm [19] operates in two search spaces: a solution space and a prioritisation space. Both searches influence each other: each solution is analysed and used to

change the prioritisation, which guides the search strategy used to find the next solution, found by restarting the search. Yokoo's Weak Commitment Search [39] (WCS) greedily constructs consistent partial assignments. On reaching a dead-end it randomly restarts, and it uses learning to maintain completeness. Richards & Richards [32] describe a SAT algorithm called learn-SAT based on WCS. None of these algorithms performs *random* backtracks on a *small* number of variables, and they therefore seem unlikely to behave as tunable stochastic local search algorithms.

An alternative hybrid approach is to augment local search with constraint propagation. Schaerf's timetabling algorithm [33], extended to constraint satisfaction problems, searches the space of *all* partial assignments (not only the consistent ones) using an objective function that includes a measure of constraint violation. Jussien & Lhomme's Path-Repair Algorithm [20] is designed for constraint satisfaction problems and applied to open shop problems. It is described as a generalisation of Schaerf's method. Besides not being restricted to consistent assignments it is different than CLS in detail: it uses learning (allowing complete versions to be devised) and it has heuristics such as clause weighting, a TABU list and greedy hill climbing. Pesant & Gendreau [27] apply systematic branch-and-bound search to efficiently explore local search neighbourhoods for scheduling and vehicle routing problems. Shaw [36] describes a vehicle routing algorithm that uses systematic search to test the legality of local search moves. Each of these algorithms permits constraint violation, implying (in the view of this author) that constraints are being under-used, which may be a drawback when solving structured problems.

There are several other hybrid approaches. Crawford [4] uses local search to select the best branching variable for a complete SAT algorithm. The two-phase algorithm of Zhang & Zhang [41] searches a space of partial variable assignments, alternating backtracking search with stochastic local search on the same data structure. It can be tuned to different problems by spending more time in either phase. A notable difference is that it *alternates* between constructive/systematic and non-constructive/non-systematic search, whereas CLS is *simultaneously* constructive/non-systematic. Kautz, McAllester & Selman's Dagsat algorithm [21] uses a special SAT problem representation (directed acyclic graphs) that makes variable dependencies explicit. Local search is then applied to the independent variables. Verfaillie & Schiex's Local Changes algorithm [37] is a complete search strategy used to solve dynamic and static constraint satisfaction problems. It extends a consistent partial assignment to a larger one by unassigning variables that are inconsistent with a new assignment, performing the new assignment then reassigning variables to the same values where possible. Experience with CLS suggests that forcing a new assignment in this way may be too disruptive for some problems.

6 Conclusion

This paper described the randomisation of the backtracking component of a systematic backtracker, with the aim of gaining scalability at the expense of completeness. The resulting algorithms can be viewed as hybrids performing stochastic local search in constrained spaces, and are therefore called Constrained Local Search (CLS) algorithms. The approach was shown to improve scalability on two very different classes of problem. A hybrid of local search and constraint propagation for SAT is the first constructive search algorithm to solve hard random 3-SAT problems with thousands of variables, and with almost identical scalability to the WSAT local search algorithm. A hybrid of local search with branch-and-bound for low-autocorrelation binary sequences (LABS) is the first stochastic search algorithm to find optimal solutions, and with improved scalability over the current best systematic backtracker. These results show that the hybridisation approach can be used to derive algorithms combining some of the best properties of stochastic and systematic search, and is applicable to both constraint satisfaction and optimisation problems.

The LABS search space topology is known to be hard for stochastic search. No stochastic algorithm has previously found optimal sequences (in the full aperiodic search space), and it has been suggested [23] that stochastic algorithms should be judged by the percentage of known optimal sequences they find. Because the CLS local search strategy is less sophisticated than most of the stochastic approaches used previously, its success on LABS is almost certainly due to the change in search space and objective function. A possible explanation is that the space of consistent partial sequences contains fewer local minima than the usual space of total sequences. The detection and characterisation of search space features detrimental to local search is a complex area of current interest (see for example [17,40]) but is beyond the scope of this paper.

This work is part of a series of studies on CLS, which has also yielded good results on graph colouring [29] and other SAT benchmarks [30]. The motivation behind CLS is to solve problems that are hard for pure stochastic search yet too large for systematic search. Its success in solving LABS problems is further evidence of its applicability to such problems. Future work will include applying CLS to skew-symmetric sequences and other combinatorial problems. Another possibility is the replacement of simple local search by evolutionary search to explore the space of partial solutions, an approach proposed in [31] for constraint satisfaction and optimisation problems. An intriguing possibility⁴ is to remove the randomness from CLS in order to make it complete without losing scalability.

Acknowledgements

The author would like to thank the anonymous referees of this paper for their extremely helpful comments.

⁴ Suggested by Ian Gent, personal communication.

References

1. R. J. Bayardo Jr, R. Schrag. Using CSP Look-Back Techniques to Solve Real World SAT Instances. *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, AAAI Press / The MIT Press 1997, pp. 203–208. 342
2. G. Beenker, T. Claasen, P. Hermens. Binary Sequences With a Maximally Flat Amplitude Spectrum. *Philips J. Res.* vol. 40, pp. 289–304. 338, 347
3. J. Bernasconi. Low Autocorrelation Binary Sequences: Statistical Mechanics and Configuration Space Analysis. *J. Physique*, vol. 48, pp. 559, 1987. 338, 347
4. J. M. Crawford. Solving Satisfiability Problems Using a Combination of Systematic and Local Search. *Second DIMACS Challenge: Cliques, Coloring, and Satisfiability*, October 1993, Rutgers University, NJ, USA. Available on: <http://www.cirl.uoregon.edu/crawford/papers/papers.html> 348
5. J. M. Crawford, L. D. Auton. Experimental Results on the Crossover Point in Random 3SAT. *Artificial Intelligence* vol. 81, nos. 1–2, March 1996, pp. 31–57. 341, 342
6. C. de Groot, D. Würtz, K. H. Hoffmann. Low Autocorrelation Binary Sequences: Exact Enumeration and Optimization by Evolutionary Strategies. *Optimization* vol. 23, Gordon & Breach Science Publishers S. A. 1992, pp. 369–384. 338, 346, 347
7. E. C. Freuder, R. Dechter, M. L. Ginsberg, B. Selman, E. Tsang. Systematic Versus Stochastic Constraint Satisfaction. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann 1995, pp. 2027–2032. 341
8. I. Gent, T. Walsh. CSPLIB: A Benchmark Library for Constraints. Available on: <http://csplib.cs.strath.ac.uk> 338, 346
9. I. P. Gent, T. Walsh. Towards an Understanding of Hill-Climbing Procedures for SAT. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI Press / MIT Press 1993, pp. 28–33. 341, 342
10. I. Gent, B. Smith. Symmetry Breaking During Search in Constraint Programming. Research Report 99.02, Department of Computer Science, University of Leeds, 1999. 338
11. M. L. Ginsberg. Dynamic Backtracking. *Journal of Artificial Intelligence Research* vol. 1, AI Access Foundation and Morgan Kaufmann 1993, pp. 25–46. 347
12. M. L. Ginsberg, D. A. McAllester. GSAT and Dynamic Backtracking. *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann 1994, pp. 226–237. 347
13. M. Golay. The Merit Factor of Long Low Autocorrelation Binary Sequences. *IEEE Transactions on Information Theory* IT-23, pp. 43–51. 337, 338
14. M. K. Goldberg, R. D. Rivenburgh. Constructing Cliques Using Restricted Backtracking. In [18] pp. 89–102. 347
15. C. Gomes, B. Selman, H. Kautz. Boosting Combinatorial Search Through Randomization. *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference*, AAAI Press / The MIT Press 1998, pp. 431–437. 347
16. W. D. Harvey. Nonsystematic Backtracking Search. PhD thesis, Stanford University, 1995. 347
17. H. Hoos. SAT-Encodings, Search Space Structure, and Local Search Performance. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, Morgan Kaufmann 1999, pp. 296–302. 349

18. D. S. Johnson, M. A. Trick (Eds). *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science* vol. 26, American Mathematical Society 1996. 341, 350
19. D. E. Joslin, D. P. Clements. Squeaky Wheel Optimization. *Journal of Artificial Intelligence Research* vol. 10, 1999, pp. 353–373. 347
20. N. Jussien, O. Lhomme. The Path-Repair Algorithm. *Proceedings of the Workshop on Large Scale Combinatorial Optimization and Constraints*, Electronic Notes in Discrete Mathematics vol. 4, 1999. Available on: <http://www.elsevier.nl/locate/disc> 348
21. H. Kautz, D. McAllester, B. Selman. Exploiting Variable Dependency in Local Search. *Poster Sessions of the Fifteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann 1997. 348
22. P. Langley. Systematic and Nonsystematic Search Strategies. *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, Morgan Kaufmann 1992, pp. 145–152. 347
23. S. Mertens. Exhaustive Search for Low-Autocorrelation Binary Sequences. *J. Phys. A: Math. Gen.* vol. 29 L473–L481, 1996. 338, 343, 349
24. S. Mertens, C. Bessenrodt. On the Ground States of the Bernasconi Model. *J. Phys. A: Math. Gen.* vol. 31, 1998, pp. 3731–3749. 338
25. P. Morris. The Breakout Method for Escaping Local Minima. *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93)*, AAAI Press / MIT Press 1993, pp. 40–45. 341
26. A. J. Parkes, J. P. Walser. Tuning Local Search for Satisfiability Testing. *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, (AAAI'96, IAAI'96)*, vol. 1, AAAI Press / The MIT Press 1996, pp. 356–362. 341
27. G. Pesant, M. Gendreau. A View of Local Search in Constraint Programming. *Principles and Practice of Constraint Programming, Proceedings of the Second International Conference*, Lecture Notes in Computer Science vol. 1118, Springer-Verlag 1996, pp. 353–366. 348
28. S. D. Prestwich. A Hybrid Local Search Algorithm for Low-Autocorrelation Binary Sequences. Technical Report TR-00-01, Department of Computer Science, University College, Cork, Ireland 2000. Available on: <http://csplib.cs.strath.ac.uk> 339
29. S. D. Prestwich. Using an Incomplete Version of Dynamic Backtracking for Graph Colouring. *Workshop on Large Scale Combinatorial Optimization*, Electronic Notes in Discrete Mathematics vol. 1, 1998. Available on: <http://www.elsevier.nl/locate/disc> 339, 340, 349
30. S. D. Prestwich. Stochastic Local Search In Constrained Spaces. *Proceedings of Practical Applications of Constraint Technology and Logic Programming, PACLP-2000*, Practical Application Company Ltd 2000, pp. 27–39. 339, 340, 341, 343, 349
31. S. D. Prestwich. A Generic Approach to Combining Stochastic Algorithms With Systematic Constraint Solvers. *Proceedings of Frontiers of Combining Systems 2 (FroCos'98)*, D. M. Gabbay and M. de Rijke (eds.), Research Studies Press/Wiley 1999, pp. 275–294. 349
32. E. T. Richards, B. Richards. Non-Systematic Search and Learning: an Empirical Study. *Principles and Practice of Constraint Programming, Proceedings of the Fourth International Conference*, Lecture Notes in Computer Science vol. 1520, Springer-Verlag 1998, pp. 370–384. 348

33. A. Schaerf. Combining Local Search and Look-Ahead for Scheduling and Constraint Satisfaction Problems. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann 1997, pp. 1254–1259. 348
34. B. Selman, H. Kautz, B. Cohen. Noise Strategies for Improving Local Search. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI Press 1994, pp. 337–343. 342, 347
35. Y. Shang, B. W. Wah. A Discrete Lagrangian-Based Global-Search Method for Solving Satisfiability Problems. *Journal of Global Optimization* vol. 10, Kluwer 1997, pp. 1–40. 342
36. P. Shaw. Using Constraint Programming and Local Search to Solve Vehicle Routing Problems. *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming (CP'98)*, Lecture Notes in Computer Science vol. 1520, Springer-Verlag 1998, pp. 417–431. 348
37. G. Verfaillie, T. Schiex. Solution Reuse in Dynamic Constraint Satisfaction Problems. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI Press 1994, pp. 307–312. 348
38. Q. Wang. Optimization by Simulating Molecular Evolution. *Biol. Cybern.* vol. 57, pp. 95–101, 1987. 338
39. M. Yokoo. Weak-Commitment Search for Solving Constraint Satisfaction Problems. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI Press 1994, pp. 313–318. 348
40. M. Yokoo. Why Adding More Constraints Makes a Problem Easier for Hill-Climbing Algorithms: Analyzing Landscape of CSPs. *Proceedings of the Third International Conference on Principles and Practice of Constraint Programming (CP'97)*, Lecture Notes in Computer Science vol. 1330, Springer-Verlag 1997, pp. 357–370. 349
41. J. Zhang, H. Zhang. Combining Local Search and Backtracking Techniques for Constraint Satisfaction. *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press / The MIT Press 1996, pp. 369–374. 348