A Hybrid Local Search Algorithm for Low-Autocorrelation Binary Sequences

Steven Prestwich
Technical Report TR-00-01
Department of Computer Science
National University of Ireland at Cork
`s.prestwich@cs.ucc.ie`

Abstract

Low-autocorrelation binary sequences have many applications in communications and electrical engineering, and their construction is a hard computational problem. Optimal sequences up to a certain length have been found by systematic search, but this approach does not scale up to large sequences. Stochastic search does scale up, but has difficulty finding optimal sequences because of the shape of the search space. This report describes a new hybrid search algorithm, combining the space-pruning ability of systematic branch-and-bound search with the scalability of stochastic search. The hybrid finds all known optimal (aperiodic) sequences, and scales better than the fastest systematic algorithm.

Chapter 1

Introduction

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

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

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 minimized. There is a well-known measure of optimality for LABS: the *merit factor* $F = N^2/2E$. Empirical curve fitting on known optimal sequences [15] yields an estimate of $F \approx 9.3$ for $N \gg 1$. LABS is of interest to the Artificial Intelligence community because of its difficulty; it is problem number 005 in the CSPLIB benchmark library [7], a web-based collection of constraint problems. Sequences with *periodic boundary conditions* can be constructed analytically for certain values of N (see [16] for example), but for the general *aperiodic* case, search is necessary. Two possibilities are *systematic* and *stochastic* search.

Systematic search usually involves the exhaustive enumeration of possibilities by backtracking, using whatever constraint programming techniques are available to eliminate redundant parts of the search space. Golay [11] used exhaustive enumeration to find optimal sequences for $N \leq 32$. Mertens [15] enumerated optimal sequences up to $N = 48$ 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 time of 68 days for exhaustive enumeration. [1] Symmetry breaking also reduces the size of the problem significantly, and Gent & Smith [9] apply 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 [9], and Mertens conjectures that for $N > 100$ progress will be made through mathematical insight rather than computer power.

[1] On a Sun UltraSparc I 170 workstation.

Where systematic search breaks down, a standard approach for solving combinatorial problems is to resort to stochastic search algorithms such as simulated annealing, evolutionary algorithms, neural networks, ant colonies or hill climbers. Such algorithms are often able to solve much larger problems. However, they perform quite poorly on some problems, and finding optimal LABS solutions is an example. Stochastic search is therefore usually applied to a restricted version of the problem: the *skew-symmetric* sequences of length $N = 2n - 1$, which satisfy $s_{n+\ell} = (-1)^\ell s_{n-\ell}$ ($\ell = 1 \ldots n-1$). This roughly halves the number of independent variables in the problem, causing a huge reduction in search space size. These sequences have been exhaustively enumerated for $N \leq 71$ [5] 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 $F \approx 6$, despite decades of research effort using simulated annealing [2, 11], evolutionary algorithms [5], analogies with molecular evolution [26] and several other heuristic search algorithms [1]. If the extrapolated value $F \approx 9.3$ is correct then either skew-symmetry excludes the most interesting large sequences, or stochastic search is unsuited to the LABS problem. In fact both explanations probably play a part: for many values of N the skew-symmetric sequences are known to be sub-optimal, and the shape of the LABS search space is considered unsuitable for stochastic search — many local minima occasionally punctuated by narrow holes containing optimal solutions, a shape referred to as a "golf-course" [15].

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 (see [6] for a debate on this issue). Hence a recent trend in Artificial Intelligence toward *hybrid* search, with the aim of combining advantages of both. Various combinations have been devised: stochastic search can be used within systematic search [3] and vice-versa [18]; the two search modes can be alternately applied to the same data structure [27]; stochastic search can be made complete by combining it with learning [10, 23]; constraint reasoning can be used to improve stochastic search [14]; and different search spaces can be explored [24].

The hybrid approach described in this report is inspired by the observation that systematic and stochastic search typically explore different spaces: the latter explores *total assignments* (in LABS terms, sequences of length N) while the former explores *consistent partial assignments* (sequences of length $\leq N$ whose extrapolated energy does not exceed an upper bound). If the space of consistent partial assignments could be explored stochastically, advantages of both might be combined: the space-pruning ability of systematic search (using techniques such as branch-and-bound and constraint propagation) and the flexibility and scalability of stochastic search. We call this approach Constrained Local Search (CLS). This report describes preliminary results showing that CLS is effective on the LABS problem. Section 2 describes CLS for LABS, Section 3 gives empirical evidence of its efficiency, and Section 4 draws conclusions.

Chapter 2

Constrained local search for LABS

The starting point for the CLS hybrid algorithm is a simplified variant of Mertens' systematic algorithm, retaining branch-and-bound but not symmetry breaking (for reasons given below).

2.1 Systematic search

First consider a depth-first search using chronological backtracking, with a constraint on the energy. 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. Initially $E_{best} = \infty$ and on finding a better solution E_{best} 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, thus pruning entire subtrees from the search tree. 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. [1] Constraint violation occurs if $E_L \geq E_{best}$.

2.2 Local search

Before CLS is described, *local search* is defined. Local search is a stochastic search method that searches 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 $\sigma \rightarrow \sigma'$

[1] This is a slightly different lower bound to that used in [15] but is based on similar reasoning.

can be classified as *backward*, *forward* or *sideways*, depending on whether $f(\sigma') - f(\sigma)$ is positive, negative or zero. A drawback of local search is that it might 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 can be applied to LABS by defining the search space as the total assignments, the objective function as the energy, and a local move as a change to the value of a single variable.

2.3 A hybrid approach

An interesting question is: what happens if backtracking is performed in a completely random way? That is, if backtracking can occur on an *arbitrary* variable? Non-systematic backtracking algorithms have been described before [12, 13], but usually as slightly modified versions of chronological backtracking. They are only non-systematic because they occasionally restart the search from a random configuration, and most backtracks are non-random. Partial-order Dynamic Backtracking (PDB) [10] does allow completely random backtracking. However, a specific aim of PDB is completeness, and to this end it uses *learning*: it accumulates information as the search proceeds, to avoid trying any assignment more than once. Although PDB is designed so that only polynomial memory in the problem size is required, it would be an advantage for large problems if learning could be avoided. Moreover, it is hard to see how PDB can exploit a technique such as branch-and-bound because it explores the space of total assignments.

The approach proposed here is to assign values to variables until a constraint violation is detected, then backtrack on a non-systematically chosen variable. The rationale is that this is also a local search algorithm: the search space is the consistent partial assignments, local moves are assigning and unassigning variables (forward and backward respectively), and the objective function to be minimized is the number of unassigned variables. This is why CLS is referred to as a hybrid. It might be objected that, under this view, *all* backtracking algorithms are hybrid local search algorithms. However, an important feature of local search is that it is *stochastic*, and if backtracking occurs on systematically-chosen variables then this is clearly not stochastic. Furthermore, it has been shown [19, 20] that, on at least some SAT problems, this greater flexibility of backtracking gives better results than slightly randomized chronological or intelligent backtracking. Not only is the choice of backtracking variable flexible: on unassigning a variable a different one may be selected for reassignment. Hence the *set* of assigned variables is also flexible.

This approach has much in common with that of Verfaillie & Schiex [25]. Their Local Changes algorithm is 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. However, Local Changes is systematic. Also, from a local search point of view, unassigning all inconsistent variables may be too disruptive for some problems. CLS also has something in common with that of Schaerf [24], who describes an algorithm that searches the space of *all* partial assignments (not only the consistent ones), using an objective function that includes a measure of constraint violation. Returning to the PDB algorithm, PDB is loosely described as a systematic version of a local search algorithm; CLS takes a complementary approach that can be seen as a non-systematic version of a systematic algorithm. More comparisons

```
procedure CLS(B):
    A = {}, U = {s_1, ..., s_N}
    while U ≠ {}
        v = VAR(U)
        D = {x ∈ {−1, 1} | assigning x to v is consistent}
        if D = {}
            for i=1 to B
                a = BACK(A), A = A − {a}, U = U ∪ {a}
        else
            assign VAL(D) to v, A = A ∪ {v}, U = U − {v}
```

Figure 2.1: The CLS schema

between CLS and other algorithms can be found in [19].

The CLS schema is shown in Figure 2.1. A is the current set of assigned variables, initialized to $\{\}$. U is the current set of unassigned variables, initialized to the full set $\{s_1, \ldots, s_N\}$. $B > 0$ is an integer parameter to the algorithm. The condition "assigning x to v is consistent" means that assigning the value x to the variable v does not violate any constraints. 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 it backtracks on B variables selected by a rule BACK. Termination is not guaranteed, but occurs if all variables are assigned ($U = \{\}$). This schema is now instantiated for LABS:

- CLS is called with $B = 1$;
- the backtracking heuristic BACK selects random assigned variables;
- the dynamic variable ordering heuristic VAR selects a random unassigned variable with probability $p = \frac{1}{N}$, and with probability $1 - p$ it selects the mostly recently unassigned variable;
- if $D = \{x\}$ then the value ordering rule VAL(D) returns x, otherwise it selects a value using a rule described below.

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 preserve consistency among sets of variables as the set of assigned variables changes, while avoiding null local moves.

The schema allows some variations not used for LABS. For example setting $B > 1$ adds noise to the search, enabling it to escape local minima. Room is also left for more sophisticated variable-ordering heuristics, and for heuristically-guided (though still non-systematic) backtracking. All these variations have been used on other problems [19, 21]. A further heuristic is added: random restart. In case CLS is trapped in a deep local minimum, it is restarted at regular intervals from a random configuration. The interval used is 1.5^N backtracks.

CLS usually requires some book-keeping to allow randomized backtracking. On the LABS problem the only book-keeping concerns the lower bound E_L on the energy. On assigning a variable v, do 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, on unassigning a variable v, do 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.

2.4 A note on symmetry breaking

This technique is 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 (approximately 8 in the case of LABS), but it also reduces the number of solutions by a similar factor, leaving the density unchanged. In fact 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 forbidden by the extra constraints.

Chapter 3

Experimental results

Optimal solutions have so far been found only by systematic search [15]. The aim of this report is to show that CLS is also capable of finding the optimal solutions, and to estimate its scalability. In future papers CLS will also be applied to large skew-symmetric sequences.

Figure 3.1 shows the results: the sequence length N, the energy E, the merit factor F, the number of backtracks taken to find the sequence, the CPU time taken [1] and the sequence itself. E_{best} was initialized to the known optimal energy plus 1 in each case. Times are mean times over various numbers of runs: 10^6 (3–6), 10^5 (7–10), 10^4 (11–15), 10^3 (16–17), 10^2 (18–20), 10^1 (21–26) and 10^0 (27–48). 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 $S = (1, 1, -1, 1, -1, -1, -1, -1, 1)$ 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 $(-1, -1, 1, 1, -1, 1, 1, 1, 1, -1)$. Sequences can have all their values inverted, or have alternate values inverted, or be reversed, without changing E. Hence sequences come in equivalence classes of size 8, apart from some that are invariant under these transformations.

The table shows that CLS is indeed able to find optimal sequences for $N \leq 48$. Plotted on a log-scale graph in Figure 3.2 and taking a least-squares fit, the estimated time taken to find a solution is $\mathrm{O}(1.68^N)$. The best systematic search takes $\mathrm{O}(1.85^N)$ to explore the *entire* search space, not simply to find the first solution (and exhaustive search without branch-and-bound takes $\mathrm{O}(2^N)$). However, the CSPLIB web site [7] contains a table of all optimal solutions for $N = 17 \ldots 50$, and in 23 out of 34 cases there is only one optimal solution (modulo symmetry); nor does the number of optimal solutions appear to increase with N. We therefore expect the time to find the first solution also to be approximately $\mathrm{O}(1.85^N)$ under systematic search, if started from random configurations. Hence CLS scales better than the best systematic algorithm.

A note on the least-squares fit. 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 may indicate that execution time is a less-than-exponential function of N, but it is not monomial because a log-log plot shows a distinct upward curve. Other researchers [8, 17] have noted a similar sub-exponential scaling for local search on hard random SAT problems, but no analytical explanation has been proposed. However, it is interesting that the same phenomenon seems to occur in LABS.

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 [5] found sequences up to $F = 6.5$ using

[1] On a 300 MHz DEC Alphaserver 1000A 5/300 under Unix.

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	6	10.08	75	0.0026	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	124512	11.1	34313131211211
28	50	7.84	394588	35.8	21211213131343
29	62	6.78	585594	58.0	212112131313431
30	59	7.63	2197440	219	132311111212155
31	67	7.17	534034	55.3	1112111122122337
32	64	8.00	457746	47.8	12212233111121117
33	64	8.51	449287	51.1	742112111111122221
34	65	8.89	6331967	712	842112111111122221
35	73	8.39	8122477	942	7122122111121111332
36	82	7.90	4376501	573	3632311131212111211
37	86	7.96	1318296	165	122221111112112448
38	87	8.30	18787115	2417	222294211211211111111
39	99	7.68	113121297	14652	23241171111141122121
40	108	7.41	18327798	2389	111211211343143131312
41	108	7.78	5234494	5070	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	13211111111124321211210212
48	140	8.23	78278179	12096	244223212113212111111811

Figure 3.1: Results on known optimal sequences

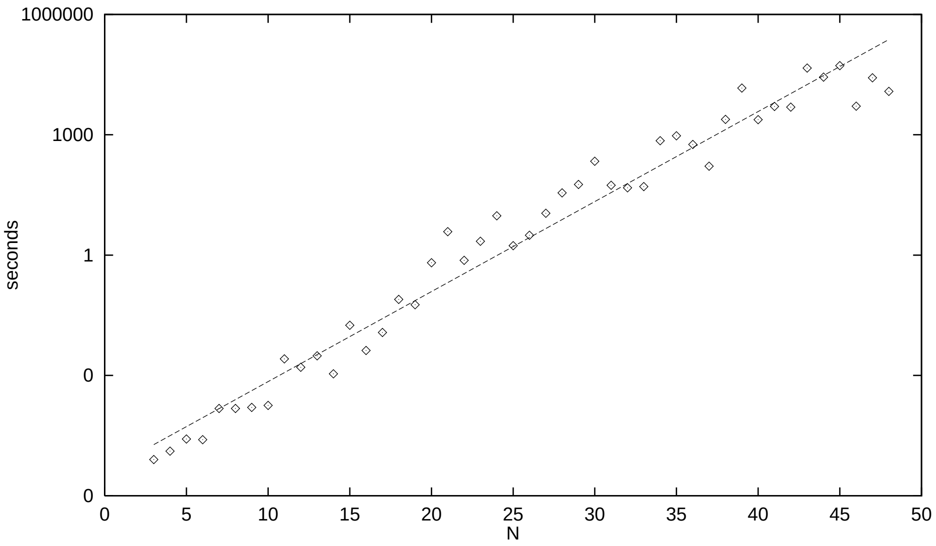

Figure 3.2: Least-squares fit on known optimal sequences

evolutionary strategies. The best result obtained by CLS after 2 days of computation was the sequence 1222122211111111B3236411211211 with $E = 262$ and $F = 7.1$. This is easily verified to be non-skew-symmetric and is the best such sequence found. However, better skew-symmetric sequences have been found by other researchers: Beenker et al [1] apply five heuristic search algorithms and find $F = 7.325$, while de Groot et al [5] find $F = 8.09$ using evolutionary strategies. Applying CLS to large skew-symmetric sequences is expected to yield good results, and this will be investigated in the near future.

Chapter 4

Conclusion

Stochastic search yields good solutions to the LABS problem by applying it to the skew-symmetric subspace. However, no stochastic algorithm has previously found *optimal* sequences in the full (aperiodic) search space. The LABS search space is considered hard for stochastic search because of its shape, and it has been suggested [15] that stochastic algorithms should be judged by the percentage of known optimal sequences they find. This report applies stochastic search to a novel search space: the consistent partial assignments. This is the space usually explored systematically, whereas stochastic search usually explores the space of total assignments. Hence the new algorithm (CLS) exploits techniques taken from both systematic and stochastic search, and is a hybrid. CLS is shown to inherit both the ability of systematic search to find optimal sequences, and the improved scalability of stochastic search. Because the stochastic algorithm used by CLS is very simple (it is similar to simulated annealing at a low fixed temperature) it is conjectured that the new search space contains fewer local minima than the usual space of total assignments.

This work is part of a series of studies on the CLS hybrid approach, which has also yielded good results on graph colouring [21] and propositional satisfiability [19, 20]. 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 replacing simple local search by evolutionary search. In fact the same space could in principle be explored by any general-purpose stochastic optimizer, an approach suggested in [22] for constraint satisfaction and optimization problems.

Bibliography

[1] G. Beenker, T. Claasen, P. Hermens. Binary Sequences With a Maximally Flat Amplitude Spectrum. *Philips J. Res.* vol. 40, pp. 289–304.

[2] J. Bernasconi. Low Autocorrelation Binary Sequences: Statistical Mechanics and Configuration Space Analysis. *J. Physique*, vol. 48, pp. 559, 1987.

[3] 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

[4] J. M. Crawford, A. B. Baker. Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI Press 1994, pp. 1092–1097.

[5] 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.

[6] 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.

[7] I. Gent, T. Walsh. CSPLIB: A Benchmark Library for Constraints. Available on: http://csplib.cs.strath.ac.uk

[8] I. P. Gent, T. Walsh. Towards an Understanding of Hill-Climbing Procedures for SAT. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington DC, USA, The AAAI Press / The MIT Press 1993, ISBN 0-262-51071-5, pp. 28–33.

[9] I. Gent, B. Smith. Symmetry Breaking During Search in Constraint Programming. Research Report 99.02, Department of Computer Science, University of Leeds, 1999.

[10] 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.

[11] M. Golay. The Merit Factor of Long Low Autocorrelation Binary Sequences. *IEEE Transactions on Information Theory* IT-23, pp. 43–51.

[12] 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.

[13] W. D. Harvey. Nonsystematic Backtracking Search. PhD thesis, Stanford University, 1995.

[14] 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.

[15] S. Mertens. Exhaustive Search for Low-Autocorrelation Binary Sequences. *J. Phys. A: Math. Gen.* vol. 29 L473–L481, 1996.

[16] S. Mertens, C. Bessenrodt. On the Ground States of the Bernasconi Model. *J. Phys. A: Math. Gen.* vol. 31, 1998, pp. 3731–3749.

[17] 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*, Portland, Oregon, vol. 1, AAAI Press / The MIT Press, 1996, ISBN 0-262-51091-X, pp. 356–362.

[18] 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.

[19] S. D. Prestwich. Stochastic Local Search In Constrained Spaces. *Proceedings of Practical Applications of Constraint Technology and Logic Programming, PACLP-2000* (to appear).

[20] S. D. Prestwich. Satisfiability Problems From Geometric Graph Colouring. *Pre-proceedings of the Tenth Irish Conference on Artificial Intelligence & Cognitive Science (AICS'99)*, University College, Cork, 1999, ISBN 0-9536826-0-9, pp. 22–28.

[21] 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

[22] 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, ISBN 0863802524, pp. 275–294.

[23] 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.

[24] 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.

[25] 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.

[26] Q. Wang. Optimization by Simulating Molecular Evolution. *Biol. Cybern.* vol. 57, pp. 95, 1987.

[27] 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.