Symmetry Breaking and Local Search Spaces

Steven Prestwich* and Andrea Roli†

*Cork Constraint Computation Centre, Department of Computer Science, University College Cork, Ireland, email: s.prestwich@cs.ucc.ie
†Dipartimento di Scienze, Università degli Studi "G.D'Annunzio", Pescara, Italy, email: a.roli@unich.it

Abstract. The effects of combining search and modelling techniques can be complex and unpredictable, so guidelines are very important for the design and development of effective and robust solvers and models. A recently observed phenomenon is the negative effect of symmetry breaking constraints on local search performance. The reasons for this are poorly understood, and we attempt to shed light on the phenomenon by testing three conjectures: that the constraints create deep new local optima; that they can reduce the relative size of the basins of attraction of global optima; and that complex local search heuristics reduce their negative effects.

1 Introduction

Symmetry-breaking has proved to be very effective when combined with complete solvers [3, 16]. This can be explained by observing that symmetry-breaking constraints considerably reduce the search space. Nevertheless, the use of symmetry-breaking constraints (hereinafter referred to as SB constraints) seem to have the opposite effect on local search-based solvers, despite the search space reduction. In [12, 13] some examples of this phenomenon are reported. When the problem is modeled with SB constraints, the search cost[1] is higher than the one corresponding to the model with symmetries.

The reasons for this phenomenon are poorly understood. Improving our understanding may aid both the modelling process and the design of future local search algorithms; in an effort to achieve this, we pose and test some conjectures. In Sec.2, we test the conjecture that SB constraints create deep new local optima, by first providing a formal example and then by showing (indirectly) empirical evidence of this phenomenon. In Secs.3 and 4, we reinforce the conjecture by exhaustively analysing small instances. Furthermore, we extend the previous conjecture by investigating more complex characteristics of the search space and we show that, in most cases, SB constraints reduce global optima reachability. We also observe that these negative effects can be tempered by using complex search strategies.

2 SB constraints and local minima

SB constraints may generate new local minima, which might also have a negative effect on local search performance. We explain the intuition behind this idea using Boolean

[1] Measured as runtime or number of variable assignments

satisfiability (SAT). The SAT problem is to determine whether a Boolean expression has a set of satisfying truth assignments. The problems are usually expressed in *conjunctive normal form*: a conjunction of clauses $\bigwedge_i C_i$ where each clause C is a disjunction of literals $\bigvee l_j$ and each literal l is either a Boolean variable x or its negation $\bar{x}$. A Boolean variable can be assigned either T (true) or F (false). A satisfying assignment has at least one true literal in each clause. Now consider the following SAT problem:

$$\overline{a} \vee b \quad \overline{a} \vee c \quad a \vee \overline{b} \quad a \vee \overline{c}$$

which is the formula $(a \leftrightarrow b) \wedge (a \leftrightarrow c)$ in conjunctive normal form. There are two solutions: [a=T, b=T, c=T] and [a=F, b=F, c=F]. Suppose that a problem modeler realises that every solution to the problem has a symmetrical solution in which all truth values are negated. Then a simple way to break symmetry is to fix the value of any variable by adding a clause such as a to the model. Denote the first model by M and the model with symmetry breaking by M_s. Now suppose we apply a local search algorithm such as GSAT [19] to the problem. GSAT starts by making a random truth assignment to all variables, then *flipping* truth assignments (changing an assignment T to F or vice-versa) to try to reduce the number of violated clauses. In model M the state [a=F, b=F, c=F] is a solution, but in M_s the added clause a is violated. Moreover, any flip leads to a state in which *two* clauses are violated: flipping a to T removes the unit clause violation but causes the first two binary clauses to be violated; flipping b [c] to T preserves the unit clause violation and also violates the third [fourth] binary clause. In other words this state has been transformed from a solution to a local minimum. (The unit clause does not preclude this as a random first state, nor does it necessarily prevent a randomized local search algorithm from reaching this state.) In contrast M has no local minima: any non-solution state contains either two T or two F assignments, so a single flip leads to a solution (respectively TTT or FFF).

GSAT (and other algorithms) will actually escape this local minimum because it makes a “best” flip even when that flip increases the number of violations, but examples can be constructed with deeper local minima. The point is that a new local minimum has been created, and local minima degrade local search performance. We can also construct examples in which propagating the SB constraints through the model still leaves a model containing new local minima. Deep local minima require a greater level of *noise* (the probability of making a move that increases the objective function being minimized) in the search algorithm, so the creation of new local minima might be indirectly detected by analysing the performance as a function of noise. We look for this phenomenon in vertex colouring problems, encoded as SAT problems. SAT is a useful form because there are a variety of publicly-available local search algorithms.

2.1 Vertex colouring as SAT

A graph $G = (V, E)$ consists of a set V of vertices and a set E of edges between vertices. Two vertices connected by an edge are said to be *adjacent*. The aim is to assign a colour to each vertex in such a way that no two adjacent vertices have the same colour. The task of finding a k-colouring for a given graph can be modeled as SAT by defining a Boolean variable for each vertex-colour combination, and adding clauses to ensure that (i) each vertex has at least one colour, (ii) no vertex has more than one colour, and (iii) no two adjacent vertices take the same colour. Colouring problems have a symmetry: the colours in any solution can be permuted. A simple but effective way of partially breaking this symmetry, used in an implementation of the DSATUR colouring algorithm [7], is to find a clique and assign a different colour to each of its vertices before search begins. Using a k-clique breaks $k!$ permutation symmetries. DSATUR uses a polynomial-time greedy algorithm to find a clique, preferring to spend time on colouring. We aim to test the effects of SB so we use a competitive clique algorithm described in [14].

We use the UBCSAT system [21] as a source of local search algorithms and experiment with three of them. Firstly the SKC (Selman-Kautz-Cohen) variant of the Walksat algorithm described in [18], which has become something of a standard algorithm in the SAT community. Its *random walk* approach was a significant advance over previous local search algorithms for SAT. However, advances in local search heuristics have been made since the invention of SKC. The Novelty and R-Novelty heuristics [6] use more sophisticated criteria for selecting variables to be flipped. These were later elaborated to the Novelty+ and R-Novelty+ variants [4] which have an extra noise parameter to avoid stagnation, and perform very well on many problems. The second algorithm we use is Novelty+. The third is SAPS (Scaling And Probabilistic Smoothing) [5] which incorporates further techniques: an efficient method for dynamically changing clause weights, an idea first used in [9] to escape local minima and since used in several algorithms; and subgradient optimization [17] inspired by Operations Research methods.

2.2 Experiments

We applied SKC to benchmark graphs from a recent graph colouring symposium. [2] On several graphs SKC moved more or less directly to a solution (using fewer steps than the problem has Boolean variables), which is quite surprising as these are considered to be non-trivial colouring problems. On such problems SB constraints often *improved* performance, counter to expectations. A typical example is shown in Fig. 1. Best performance is obtained with high noise, showing that a simple random walk algorithm finds the problem trivial. Adding SB constraints seems to give the algorithm a head start, transforming a trivial problem into an even more trivial one. Some other graphs gave similar results, including the mulsol.i.n graphs.

The timetabling graph school1 is nontrivial but not very hard for SKC, taking about twice as many flips as there are Boolean variables. But adding SB constraints makes the problem at least 4 orders of magnitude harder in most SKC runs, (though performance is better with frequent random restarts). Novelty+ also finds this problem extremely hard with SB constraints. SAPS finds the problem trivial without SB constraints, solving it in fewer flips than there are Boolean variables. With SB constraints it takes several times longer but is far more robust than SKC and Novelty+.

It turns out that many colouring benchmarks are either too trivial to show the effects we are looking for (like anna) or too intractable to study in detail (like school1). We

[2] http://mat.gsia.cmu.edu/COLOR04

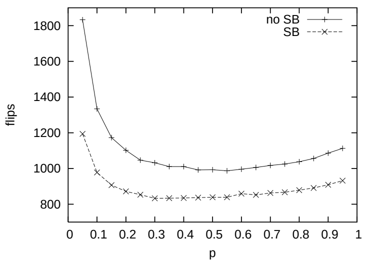

Fig. 1. SKC results for the anna graph

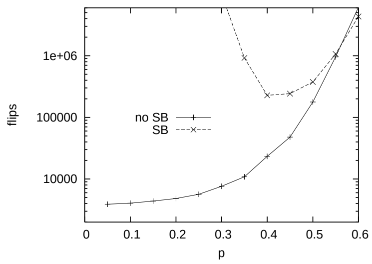

Fig. 2. SKC results for the flat graph

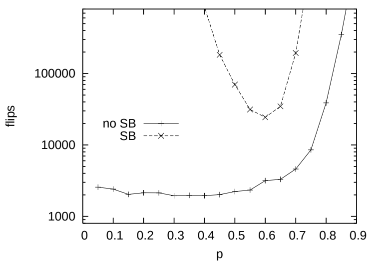

Fig. 3. Novelty+ results for the flat graph

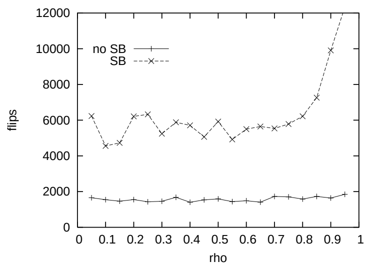

Fig. 4. SAPS results for the flat graph

therefore created our own graphs using J. Culberson's graph generator. [3] We chose a flat graph, which is a randomly generated graph containing a hidden colouring. We chose an edge density of 0.5, flatness 0, 100 vertices, with a hidden 10-colouring, and found an 8-clique. The results are shown in Fig. 2 (each data point is the median over 100 runs) and confirm the effect we are looking for. Without SB constraints the problem is non-trivial, taking several times more flips than there are Boolean variables. Best results are obtained with low noise. With SB constraints the problem is about 2 orders of magnitude harder using optimal noise, which is higher than without SB constraints. In case this result is an artefact of SKC's heuristics we repeated the experiment with Novelty+, shown in Figure 3. This algorithm is often more efficient than SKC but the same pattern emerges. We believe that this indicates the increased ruggedness of the search space, caused by new local minima.

Next we tried SAPS on the same graph, which has 3 parameters besides the usual noise parameter p. SAPS performance is reported to be robust with respect to the default values of p and two of the other parameters, but less so for the smoothing parameter ρ. The greater the value of ρ the more rapidly recent history is forgotten, and the less likely the algorithm is to escape from a local minimum. Thus ρ can be viewed as a form of inverse noise parameter. Fig. 4 shows the results for the flat graph, this time varying ρ. Without SB constraints the performance is independent of ρ. With SB constraints

[3] http://web.cs.ualberta.ca/~joe/Coloring/index.html

performance is worse, especially for high ρ values, but it is much more robust than SKC or Novelty+. This is still consistent with our conjecture that SB constraints add new local minima. It also suggests that complex local search, with heuristics designed to escape local minima, are less affected by SB constraints — though they are still adversely affected.

The above examples show that SB constraints can have a huge impact on local search performance. These results are clearer and more extreme than those in [12, 13]. We conjectured that the cause is the creation of new, deep local minima, and our results support this conjecture but do not provide direct evidence. Ideally, we should analyse the search spaces of colouring problems, for example to count the number of local optima with and without SB constraints. Unfortunately, the search spaces are too large for an exhaustive enumeration, but in the following we analyse the search space of a hard optimization problem suitable for this kind of study. Before this, we formally define the search space explored by local search.

3 The search graph and its main characteristics

The number of local minima can be taken as an indicator of the *ruggedness* of the search space explored by local search. In turn, it is usually recognized that the more rugged a search space is, the poorer is local search performance. Nevertheless, this parameterization of the search space might be sometimes insufficiently explicative or predictive. In this section, we provide a simple model of the search space which not only considers local and global optima, but also their *basins of attraction*, i.e., the set of states from which the optima can be reached.

The local search process can be viewed as an exploration of a landscape aimed at finding an optimal solution, or a *good* solution, i.e., a solution with a quality above a given threshold.[4] We define the search space explored by a local search algorithm as a *search graph*. The topological properties of such a graph are defined upon the neighborhood structure, that generate the *neighborhood graph*.

3.1 Neighborhood and search graphs

A *Neighborhood Graph* (NG), is defined by a triple: $\mathcal{L} = (S, \mathcal{N}, f)$, where:

- S is the set of feasible states;
- $\mathcal{N}$ is the neighborhood function $\mathcal{N} : S \rightarrow 2^S$ that defines the neighborhood structure, by assigning to every $s \in S$ a set of states $\mathcal{N}(s) \subseteq S$.
- f is the objective function $f\colon S \rightarrow \mathbb{R}^+$

The neighborhood graph can be interpreted as a graph (see Fig. 5) in which nodes are states (labeled with their objective value) and arcs represent the neighborhood relation between states. The neighborhood function $\mathcal{N}$ implicitly defines an *operator* φ which takes a state s_1 and transforms it into another state $s_2 \in \mathcal{N}(s_1)$. Conversely, given an operator φ, it is possible to define a neighborhood of a variable $s_1 \in S$:

[4] For the rest of this paper, we will suppose, without loss of generality, that the goal of the search is to find an optimal solution. Indeed, the same conclusions we will draw can be extended to a set including also good solutions.

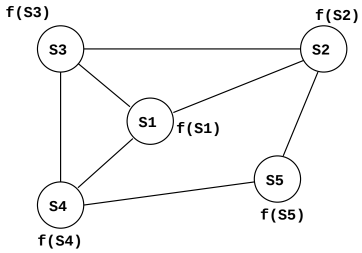

Fig. 5. Example of undirected graph representing a neighborhood graph (fitness landscape). Each node is associated with a solution s_i and its corresponding objective value $f(s_i)$. Arcs represent transition between states by means of φ. Undirected arcs correspond to symmetric neighborhood structure.

$$\mathcal{N}_\varphi(s_1) = \{s_2 \in S \setminus \{s_1\} \mid s_2 \text{ can be obtained by one application of } \varphi \text{ on } s_1\}$$

In most cases, the operator is *symmetric*: if s_1 is a neighbor of s_2 then s_2 is a neighbor of s_1. In a graph representation (like the one depicted in Fig. 5) undirected arcs represent symmetric neighborhood structures. A desirable property of the neighborhood structure is to allow a path from every pair of nodes (i.e., the neighborhood is strongly optimally connected) or at least from any node to an optimum (i.e., the neighborhood is weakly optimally connected). Nevertheless, there are some exceptions of effective neighborhood structures which do not enjoy this property [10].

The exploration process of local search methods can be seen as the evolution in (discrete) time of a discrete dynamical system [1]. The algorithm starts from an initial state and describes a trajectory in the state space, that is defined by the neighborhood graph. The system dynamics depends on the strategy used; simple algorithms generate a trajectory composed of two parts: a *transient* phase followed by an *attractor* (a fixed point, a cycle or a complex attractor). Algorithms with advanced strategies generate more complex trajectories which can not be subdivided in those two phases.

It is useful to define the search as a walk on the neighborhood graph. In general, the choice of the next state is a function of the search history (the sequence of the previously visited states) and the iteration step. Formally: $s(t+1) = \phi(\langle s(0), s(1), \ldots, s(t)\rangle, t)$ where the function ϕ is defined on the basis of the search strategy. ϕ could also depend on some parameters and can be either deterministic or stochastic.

For instance, let us consider a deterministic version of the Iterative Improvement local search. The trajectory starts from a point s_0, exhaustively explores its neighborhood, picks the neighboring state s' with minimal objective function value[5] and, if s' is better than s_0, it moves from s_0 to s'. Then this process is repeated, until a minimum $\hat{s}$ (either local or global) is found. The trajectory does not move further and we say that the system has reached a fixed point ($\hat{s}$). The set of points from which $\hat{s}$ can be reached is the basin of attraction of $\hat{s}$. Note that, especially in the case of deterministic local search algorithms, not every initial state is guaranteed to reach a global optimum.

[5] Ties are broken by enforcing a lexicographic order of states.

Once we have introduced also the search strategy, the edges of the neighborhood graph can be oriented and labeled with transition probabilities (whenever it is possible to evaluate them). This will lead to the definition of concepts such as basins of attraction, state reachability and graph navigation. In the following, this resulting graph will be referred to as the *search graph*.

3.2 Basins of attraction

The concept of *basin of attraction* (BOA) has been introduced in the context of dynamical systems, in which it is defined referring to an *attractor*. Concerning our model of local search, we will use the concept of basin of attraction of any node of the search graph. For the purposes of this paper we only consider the case of deterministic systems, even if it is possible to extend the definition to stochastic ones.[6]

Definition Given a deterministic algorithm $\mathcal{A}$, the basin of attraction $\mathcal{B}(\mathcal{A}|s)$ of a point s, is defined as the set of states that, taken as initial states, give origin to trajectories that include point s. The cardinality of a basin of attraction represents its size (in this context, we always deal with finite spaces).

Given the set S_{opt} of the global optima, the union of the BOA of global optima $I_{opt} = \bigcup_{i \in S_{opt}} \mathcal{B}(\mathcal{A}|i)$ represents the set of desirable initial states of the search. Indeed, a search starting from $s \in I_{opt}$ will eventually find an optimal solution. Since it is usually not possible to construct an initial solution that is guaranteed to be in I_{opt}, the ratio *rGBOA* = $|I_{opt}|/|S|$ can be taken as an indicator of the probability to find an optimal solution. In the extreme case, if we start from a random solution, the probability of finding a global optimum is exactly $|I_{opt}|/|S|$. Therefore, the higher this ratio, the higher the probability of success of the algorithm.

Given a local search algorithm $\mathcal{A}$, the topology and structure of the search graph determine the effectiveness of $\mathcal{A}$. In particular, the reachability of optimal solutions is the key issue. Therefore, the characteristics of the BOA of optimal solutions are of dramatic importance.

4 SB constraints and basins of attraction

We now analyse the search space of an optimization problem to examine whether the number of local minima is indeed increased. We also test another conjecture: that SB constraints harm local search by reducing *rGBOA* defined on the basis of a simple iterative improvement local search. In fact, even the most complex local search algorithms incorporate a greedy heuristic which is the one that characterizes iterative improvement. Therefore, if SB constraints reduce *rGBOA*, then the more a local search is similar to iterative improvement, the more it should be affected by SB constraints. Furthermore, we should also observe that local search algorithms equipped with complex exploration

[6] In this case, not only probabilistic basins of attraction are of interest, but also the probability a given state can be reached. Some studies in this direction are subject of ongoing work.

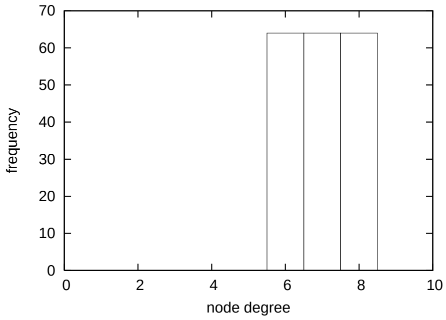

Fig. 6. Node degree frequency of the neighborhood graph in the case of the model with SB constraints ($n = 10$).

strategies are less affected by SB constraints. In this work, we aim at experimentally verifying this conjecture. By relating local search performance with *rGBOA* we consider a more general case than only counting local optima.[7]

We test our conjectures on the LABS optimization problem, which is to find an assignment to binary variables such that an energy function defined upon them is minimized. Given n binary variables $x_1, \ldots, x_n$, which can assume a value in $\{-1, +1\}$, we define the k-th correlation coefficient of a complete variable assignment $s = \{(x_1 = d_1), \ldots, (x_n = d_n)\}$ with $d_i \in \{-1, +1\}, i = 1, \ldots, n$, as $C_k(s) = \sum_{i=1}^{n-k} x_i \; x_{i+k}$, $k = 1, \ldots, n-1$ and the total function to be minimized is $E(s) = \sum_{k=1}^{n-1} C_k^2(s)$.

4.1 Analysis of the space

We exhaustively explored the search space of LABS, for n ranging from 6 up to 18.[8] As neighborhood function we chose the one defined upon unitary Hamming distance, that is the most used one for problems defined over binary variables. (We emphasize that this choice determines the fundamental topological properties of the search space.)

In the model with SB constraints, only a subset of symmetric solutions has been cut, by enforcing constraints on the three left-most and right-most variables [8]. In these experiments the SB constraints are *enforced* instead of used to modify the objective function. In the SAT examples SB constraints were handled like any other clauses: when they were violated they increased the objective function being minimized (the number of violated clauses). In our LABS experiments the SB constraints are never violated so the local search space is smaller. Our experiments are therefore also a test of whether the observed negative effects still occur when SB constraints are used to restrict the search space.

It is first interesting to study how the neighborhood graph changes upon the application of SB constraints. In M, the neighborhood graph induced by single variable

[7] Indeed, *rGBOA* can be decreased even if the density of local optima is not increased.

[8] The size limit is due to the exhaustiveness of the analysis.

Table 1. Search space characteristics of LABS instances.

n	feasible states		global optima		local optima		global BOA	
	no SB	with SB	no SB	with SB	no SB	with SB	no SB	with SB
6	64	12	28	5	0	0	1.0	1.0
7	128	24	4	1	24	5	0.40625	0.33333
8	256	48	16	3	8	2	0.86328	0.77083
9	512	96	24	4	84	16	0.42969	0.37500
10	1024	192	40	7	128	29	0.54590	0.45833
11	2048	384	4	1	240	52	0.03906	0.04427
12	4096	768	16	3	264	61	0.07544	0.06901
13	8192	1536	4	1	496	111	0.01831	0.01953
14	16384	3072	72	11	664	177	0.21240	0.15202
15	32768	6144	8	2	1384	326	0.01956	0.01742
16	65536	12288	32	8	1320	332	0.05037	0.04972
17	131072	24576	44	9	3092	721	0.05531	0.04073
18	262144	49152	16	2	5796	1372	0.02321	0.01068

flips is a hypercube in which each node is connected to n other nodes. This graph has a constant degree equal to n. The neighborhood graph associated to M_s is characterized by a node degree frequency that varies in a small range, around a mean value slightly smaller than n (see an example in Fig. 6). The topological characteristics of this graph are not affecting the search, since the reachability of nodes is not significantly perturbed. Therefore, we can exclude that SB constraints in LABS affect local search by perturbing the topological properties of the neighborhood graph.

We consider now the features of the search graph, which, in general, can be algorithm-dependent. (This is the case for basins of attraction, while local and global optima only depend on the objective function and the neighborhood.) The search space characteristics of interest are the number of feasible states, the number of global and local optima and the value *rGBOA*. These values are reported in Tab.1. We first observe that the search space reduction yielded by applying SB constraints is 5.33, independently of the instance size, while the global optima ratio is on average 5.19 (std.dev. is 1.22) and the local optima ratio is 4.36 (std.dev. is 0.40). Therefore, the number of local optima is reduced by a factor which is less than the search space reduction. An interesting perspective of the search space can be given by plotting the global (resp. local) optima density, i.e., the ratio of the number of global (resp. local) optima to the search space size. The density of optima is plotted in Figs. 7 and 8. These plots show that the density of local optima is always higher in the model M_s, proving that SB constraints increase the search space ruggedness. By enumerating the whole search space, we also observed that *new* local optima are created in M_s: some feasible states in M_s are local optima in M_s, but not in M. Note also that the density of global optima decreases exponentially with n. (The relation between number of global optima and n can be fitted with a good approximation by a line in a semi-logarithmic plot.) On the other side, the density of

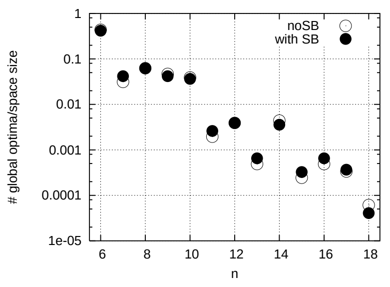

Fig. 7. Ratio of global optima w.r.t. search space size for $n = 6, \ldots, 18$, in a semi-log plot.

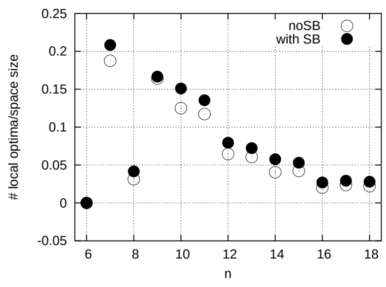

Fig. 8. Ratio of local optima w.r.t. search space size for $n = 6, \ldots, 18$.

local optima decreases much more slowly for the highest values of n. This provides an explanation for LABS being particularly difficult for local search.

Finally, we consider the basins of attraction of global optima. The basins of attraction are defined with respect to deterministic iterative improvement. We note that in all the cases, except for $n = 11$ and $n = 13$, *rGBOA*(M_s)<*rGBOA*(M).

4.2 Experiments

We attacked LABS (with $n = 6, \ldots, 18$) with four different local search algorithms: Best improvement with randomly broken ties (BI), First improvement with random order among neighbors (FI), Simulated annealing (SA) and Tabu search (TS).[9] From the perspective of search space exploration, the algorithms chosen exhibit a varying explorative attitude, starting from the lowest of BI to the highest of TS, while all keeping a 'greedy' character.[10] We run each algorithm on the original model and on the model with SB constraints. The algorithms are stopped after $10n$ non-improving moves. This termination condition enables us to compare the algorithms on the basis of the best solution they returned once a steady state is reached. (In the literature of metaheuristics, this state is also commonly called *stagnation*.) Tab.2 gives a synoptic view of the algorithm performance in term of success ratio (out of 1000 runs).

A graphic comparison of the performance of each algorithm on the two problem models is given in Figs. 9, 10, 11, 12, in which we plotted the difference of solved instances (perc.) against n, i.e., $\Delta_{\%} = 100 \times (\text{solved(noSB)}-\text{solved(SB)})/1000$. Note that the performance on M dominates the one on M_s in all but the TS case.

The correlation between number of successes and *rGBOA* is particularly interesting. From the plots in Figs. 13, 14, 15 and 16, we observe that for BI and FI the number of

[9] The initial temperature in SA has been set after a simple trial-and-test procedure. The tabu tenure in TS is randomly restarted each iteration in a range between 1 and $n/2$, in the spirit of robust tabu search [20].

[10] A deep discussion on this topic, involving also intensification and diversification, can be found in [2].

Table 2. Synopsis of the number of solved instances (out of 1000 runs) of the four local search algorithms on the original model and the model with SB constraints.

n	BI		FI		SA		TS	
	no SB	with SB	no SB	with SB	no SB	with SB	no SB	with SB
6	1000	1000	1000	1000	1000	1000	1000	1000
7	417	324	530	484	1000	999	1000	1000
8	875	751	825	728	1000	960	1000	1000
9	438	342	266	267	913	834	1000	1000
10	561	426	995	720	996	920	1000	1000
11	42	41	47	30	101	136	528	928
12	77	63	39	37	308	318	835	895
13	16	17	3	5	105	111	283	251
14	200	125	202	142	857	731	992	1000
15	25	24	22	20	157	179	363	599
16	66	52	29	39	336	307	819	919
17	60	43	71	45	508	388	909	916
18	25	6	35	8	270	131	678	412

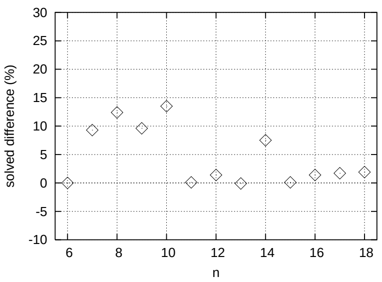

Fig. 9. Best improvement. Percentage of the difference between solved instances in the original model and solved instances in the model with SB constraints.

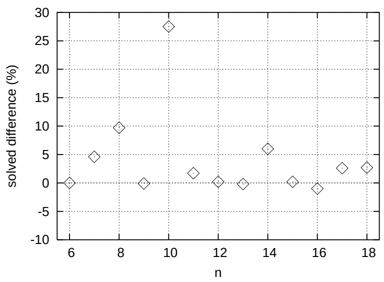

Fig. 10. First improvement. Percentage of the difference between solved instances in the original model and solved instances in the model with SB constraints.

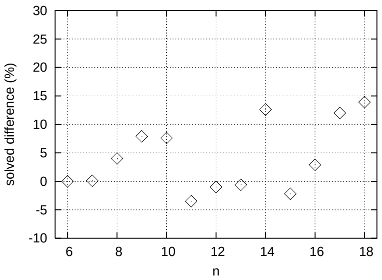

Fig. 11. Simulated annealing. Percentage of the difference between solved instances in the original model and solved instances in the model with SB constraints.

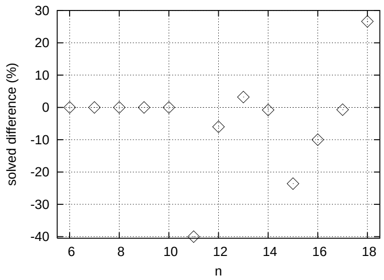

Fig. 12. Tabu search. Percentage of the difference between solved instances in the original model and solved instances in the model with SB constraints.

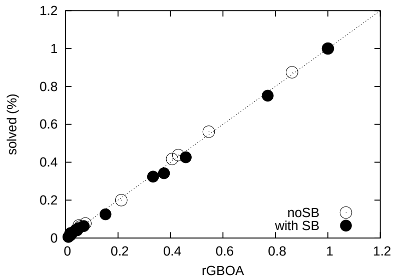

Fig. 13. Best improvement. Solved instances (perc.) plotted against the size of the global optima BOA.

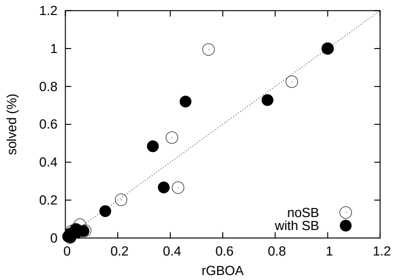

Fig. 14. First improvement. Solved instances (perc.) plotted against the size of the global optima BOA.

successes is proportional to the size of the global optima basin of attraction. In the case of SA and TS, while the correlation is still observable, we note that the performance remains quite high even for low values of *rGBOA*, especially in the case of TS. The value of *rGBOA* have been measured on the basis of deterministic best improvement, therefore it is not surprising that both BI and FI show a proportional relation between successes and fraction of states that make the search converging to a global optimum. SA performs a more effective search space exploration than iterative improvement procedures, and even more TS, therefore the number of successes they achieve is much higher than that of BI and FI. It is interesting to note that the performance of both SA and TS starts to degrade (quite abruptly) when the normalized size of the global optima basin of attraction approaches a threshold value. Moreover, for TS this value is smaller than for SA, in other words, the more sophisticated an exploration strategy is, the lower the value of *rGBOA* at which the performance starts to be strongly affected.

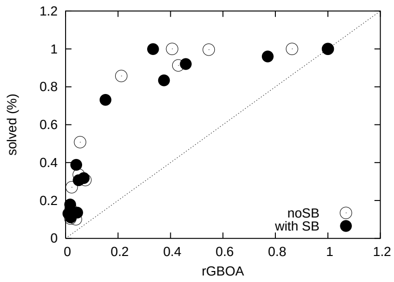

Fig. 15. Simulated annealing. Solved instances (perc.) plotted against the size of the global optima BOA.

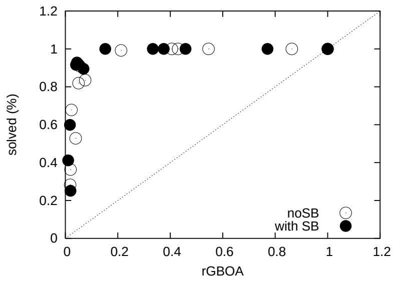

Fig. 16. Tabu search. Solved instances (perc.) plotted against the size of the global optima BOA.

4.3 Discussion

The available data are still not sufficient to draw strong conclusions, but we have experimental results to support our conjectures. First of all, we have directly observed that new local optima are created and their density is higher in M_s. This is one of the reasons why *rGBOA* is reduced in the model with SB constraints. Another important observation is that local search performance is strongly affected by the size of the global optima basin of attraction. This relation is in the form of a positive correlation (i.e., the smaller the BOA, the lower the performance) and it is well approximated by a linear relation in the case of simple local search algorithms (BI and FI), while it is nonlinear in the case of more complex search strategies (SA and TS). The nonlinearity of this relation plays a big role when we compare the performance of local search algorithms (in terms of success ratio). In fact, large differences in *rGBOA* imply large deviations of the performance. But on the other side, when the difference is quite small, other factors come into play.

It is important to note that these effects occur even when SB constraints are used to *restrict* the search space. In the SAT experiments the search spaces with and without SB constraints were the same size: the SB constraints merely modified the objective function (number of violated constraints). It was therefore possible for the search to approach an excluded solution that has become a local minimum. But in the LABS experiments this was not possible: the search never strayed into the excluded subspaces, yet the SB constraints still had negative effects on local search performance. This is perhaps more surprising than the SAT results.

5 Conclusion

We showed, both empirically and by analysis, that SB constraints have two distinct negative effects on a local search space: they increase the relative size of local optima, and they reduce the relative size of global basins of attraction. These effects were observed using two different symmetry breaking methods: using SB constraints to modify an objective function, and using them to restrict the search space. The effects can be extremely strong, slowing down local search performance by several orders of magnitude. However, the effects are reduced by using more complex local search algorithms with heuristics for escaping local minima and with diversification strategies.

We believe that more research into both modelling and local search is necessary. Several lines of research are possible. Firstly, it has been suggested that, because SB constraints can have such a strong effect on local search, performance might be greatly improved by *adding* symmetry [13, 15]. We have so far found only one application of this idea (Golomb rulers [13]) but the technique seems promising. Secondly, there is a startling difference in performance between local search algorithms with and without mechanisms for escaping local minima, and our problems with SB constraints provide a useful test for such mechanisms. By focusing on such problems we might discover even better mechanisms, such as effective weighting procedures for SB constraints. (These mechanisms may include also the use of SB constraints in a TS fashion and the iterative relaxation of some of them during the search.) Thirdly, we still lack a general model (or, at least, general criteria) to both explain the phenomenon and guide the modelling phase. For example, it would be important to know if there are specific classes of SB constraints which induce a search space that is particularly unfavourable for exploration by local search. Fourthly, the effects another symmetry breaking technique have yet to be explored: *reformulation*, that is formulating a new model of the problem that does not contain the symmetry. Negative effects here would be even more surprising, and have important implications for problems modeling.

Finally, it should be noted that these negative effects are caused by symmetry *breaking*, but symmetry can be exploited in other ways to improve local search performance [11].

Acknowledgment

The authors thank the anonymous reviewers for useful comments and suggestions. This work has received support from Science Foundation Ireland under Grant 00/PI.1/C075.

References

1. Y. Bar–Yam. *Dynamics of Complex Systems*. Studies in nonlinearity. Addison–Wesley, 1997.
2. C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, September 2003.
3. James Crawford, Matthew L. Ginsberg, Eugene Luck, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *KR'96: Principles of Knowledge Representation and Reasoning*, pages 148–159. Morgan Kaufmann, San Francisco, California, 1996.
4. H. H. Hoos. On the run-time behaviour of stochastic local search algorithms. In *Sixteenth National Conference on Artificial Intelligence*, pages 661–666. AAAI Press, 1999.
5. F. Hutter, D. A. D. Tompkins, and H. H. Hoos. Scaling and probabilistic smoothing: Efficient dynamic local search for sat. In *Eighth International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science*, pages 233–248. Springer, 2002.

6. D. A. McAllester, B. Selman, and H. A. Kautz. Evidence for invariants in local search. In *Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, pages 321–326. AAAI Press / MIT Press, 1997.
7. A. Mehrotra and M. A. Trick. A column generation approach to graph colouring. *INFORMS Journal on Computing*, 8:344–354, 1996.
8. S. Mertens. Exhaustive search for low-autocorrelation binary sequeneces. *J. Phys. A*, 29:L473–L481, 1996.
9. P. Morris. The breakout method for escaping from local minima. In *Eleventh National Conference on Artificial Intelligence*, pages 40–45. AAAI Press / MIT Press, 1993.
10. E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job-shop problem. *Management Science*, 42(2):797–813, 1996.
11. A. Petcu and B. Faltings. Applying interchangeability techniques to the distributed breakout algorithm. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2003.
12. S. Prestwich. First-solution search with symmetry breaking and implied constraints. In *Proc. of CP'01 Workshop on Modelling and Problem Formulation*, 2001.
13. S. Prestwich. Supersymmetric modeling for local search. In *Proc. of SymCon'02 Workshop on Symmetry and Constraint Satisfaction Problems*, 2002.
14. S. D. Prestwich. Combining the scalability of local search with the pruning techniques of systematic search. *Annals of Operations Research*, 115:51–72, 2002.
15. S. D. Prestwich. Negative effects of modeling techniques on search performance. *Annals of Operations Research*, 118:137–150, 2003.
16. J. F. Puget. Symmetry breaking revisited. In *Principle and Practice of Constraint Programming – CP02*, volume 2470 of *Lecture Notes in Computer Science*, pages 446–451. Springer-Verlag, 2002.
17. D. Schuurmans, F. Southey, and R. C. Holte. The exponentiated subgradient algorithm for heuristic boolean programming. In *Seventeenth International Joint Conference on Artificial Intelligence*, pages 334–341. Morgan Kaufmann Publishers, 2001.
18. B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Twelfth National Conference on Artificial Intelligence*, pages 337–343. AAAI Press, 1994.
19. B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446. MIT Press, 1992.
20. E. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.
21. D. A. D. Tompkins and H. H. Hoos. Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat. In *Seventh International Conference on Theory and Applications of Satisfiability Testing*, pages 37–46, 2004.