Local Optima Network Analysis for MAX-SAT

Gabriela Ochoa
University of Stirling
Stirling, Scotland, UK
gabriela.ochoa@cs.stir.ac.uk

Francisco Chicano
University of Malaga
Malaga, Spain
chicano@lcc.uma.es

ABSTRACT

Local Optima Networks (LONs) are a valuable tool to understand fitness landscapes of optimization problems observed from the perspective of a search algorithm. Local optima of the optimization problem are linked by an edge in LONs when an operation in the search algorithm allows one of them to be reached from the other. Previous work analyzed several combinatorial optimization problems using LONs and provided a visual guide to understand why the instances are difficult or easy for the search algorithms. In this work we analyze for the first time the MAX-SAT problem. Given a Boolean formula in Conjunctive Normal Form, the goal of the MAX-SAT problem is to find an assignment maximizing the number of satistified clauses. Several random and industrial instances of MAX-SAT are analyzed using Iterated Local Search to sample the search space.

CCS CONCEPTS

• **Mathematics of computing → Combinatorial optimization**;
• **Theory of computation → Random search heuristics**;

KEYWORDS

Local Optima Networks, MAX-SAT, Combinatorial Optimization, Funnels

ACM Reference Format:
Gabriela Ochoa and Francisco Chicano. 2019. Local Optima Network Analysis for MAX-SAT. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3319619.3326855

1 INTRODUCTION

Local Optima Networks (LONs) have proven to be a valuable tool to analize and visualize the search landscape of combinatorial optimization problems [5, 10]. A LON is a graph where the set of vertices are local optima of an optimization problem and an edge between two nodes exists if there is a way to "jump" from one to the other. Depending on which relationship the edges represent we have different kinds of LONs. In this paper, two local optima x and y will be connected by an edge if it is possible to reach y from x after perturbing x and applying a hill climber.

Local Optima Networks have been computed and analyzed for many combinatorial problems, including Quadratic Assignment [3], Permutation Flow-Shop [4], Traveling Salesman [6], and Number Partitioning [7]. Several different local optima structures have been identified in the LONs of the studied fitness landscapes, including plateaus and funnels. Their presence (or absence) serves to explain the performance of trajectory-based methods such as Iterated Local Search on the underlying optimization problems. And LONs are a useful tool to analyse the global structure of fitness landscapes.

MAX-SAT is the optimization version of the Satisfiability (SAT) problem. Given a Boolean formula, SAT checks if there is an assignment of variables to Boolean values such that the formula is satisfiable. The Boolean formula is commonly expressed as a conjunction of clauses (Conjunctive Normal Form). A clause is a list of literals (a Boolean variable or its negated) that is satisfied if at least one literal is true. The Boolean formula is satisfiable if all the clauses are. The goal of MAX-SAT is to find an assignment to maximize the number of satisfied clauses. Thus, the objective function (to be maximized) is the number of satisfied clauses.

To the best of our knowledge the LON model has not been applied before to the MAX-SAT problem. Indeed, MAX-SAT has proven difficult to model with LONs in the past, due to the existence of large plateaus in the underlying search space. An analysis with LONs can shed light onto the structure of MAX-SAT fitness landscapes, which can help to increase our understanding and improve the design and selection of optimization algorithms.

The organization of the paper is as follows. In Section 2 the basic concepts of Local Optima Networks are introduced. Section 3 introduces Gray Box Optimization and the hill climber used in our Iterated Local Search algorithm. Sections 4 and 5 present the methodology used in the experimental evaluation and the results obtained. The paper concludes with Section 6.

2 LOCAL OPTIMA NETWORKS

A recent variant of LONs, the Compressed Monotonic LONs (CMLONS) [7], allows modelling the funnel structure of landscapes with neutrality (i.e. existence of plateaus of local optima with equal fitness). We describe below the LON model, before introducing the Compressed Monotonic model (CMLON).

2.1 LON Model

A fitness landscape [8] is a triplet (S, N, f) where S is a set of potential solutions i.e., a search space, $N : S \longrightarrow 2^S$, a neighbourhood structure, is a function that assigns to every $s \in S$ a set of neighbours $N(s)$, and $f : S \longrightarrow \mathbb{R}$ is a fitness function that can be pictured as the *height* of the corresponding solutions.

In our study, the search space is $\mathbb{B}^N$, i.e. the space of binary strings of length N, so its size is 2^N. As neighbourhood, we consider

the standard Hamming distance 1 neighbourhoods, that is, the set of all solutions at most Hamming distance 1, respectively, from the current solution.

LON. Is the directed graph LON = (L, E), where L is the set of the local optima, and E is the set of escape (perturbation) edges.

Local optima. A local optimum, which in MAX-SAT is a maximum, is a solution l such that $\forall s \in N(l)$, $f(l) \geq f(s)$. Notice that the inequality is not strict, in order to allow the treatment of neutrality (local optima of equal fitness), which is known to widely occur on MAX-SAT. The set of local optima, which corresponds to the set of nodes in the network model, is denoted by L. Local optima are determined with the efficient local search algorithm described in Section 3.2.

Perturbation edges. Edges are directed and based on the perturbation operator (k-bitflips). There is an edge from local optimum l_1 to local optimum l_2, if l_2 can be obtained after applying a random perturbation (k-bitflips) to l_1 followed by local search. Edges are weighted with estimated frequencies of transition. We determined the edge weights in a sampling process. The weight is the number of times a transition between two local optima occurred. The set of edges is denoted by E.

2.2 Compressed Monotonic LON Model

Compressed local optima. A compressed local optimum (also called a local optima plateau) is a set of connected nodes in the LON with the same fitness value. Two nodes are connected if there is an edge between them. The set of compressed optima with the same fitness, denoted by CL, corresponds to the set of nodes in the Compressed Monotonic LON model.

Monotonic Perturbation edges. The set of perturbation edges E as defined for the LON model above, but after removing deteriorating edges, that is, edges connecting a pair of nodes where the end node has inferior fitness than the start node. We call this set monotonic, ME, as it contains only non-deteriorating transitions between optima.

Compressed Monotonic LON. Is the directed graph CMLON = (CL, CE), where nodes are the compressed local optima CL. The edges CE are aggregated from the monotonic edge set ME by summing up the edge weights.

Monotonic Sequence. A monotonic sequence is a path of connected nodes $MS = \{cl_1, cl_2, \ldots, cl_s\}$ where $cl_i \in CL$. By definition of the edges, $f(cl_i) \geq f(cl_{i-1})$. There is a natural end to every monotonic sequence, cl_s, when no improving transitions can be found. In the directed CMLON network, cl_s will be a node without outgoing edges (called a sink in the graph theory terminology).

Funnel. A funnel can be loosely described a grouping of local optima, conforming a coarse-grained gradient towards a high fitness optimum. More formally, we characterise funnels in the CMLON as the aggregation of all monotonic sequences ending at the same point (funnel top or sink). Funnels can be seen as basins of attraction at the level of local optima.

3 GRAY BOX OPTIMIZATION

We will work along the paper with functions defined over a set of Boolean variables x_i, each one taking values 0 and 1. We say that a function f of n variables has k-bounded epistasis if it can be written as a sum of m subfunctions f_l, each one depending on at most k variables:

$$f(x) = \sum_{l=1}^{m} f_l(x_{i_{l,1}}, x_{i_{l,2}}, \ldots, x_{i_{l,k}}), \tag{1}$$

where $i_{l,j}$ is the index of the j-th variable in subfunction f_l. In the case of binary variables, these functions have been named Mk Landscapes by Whitley et al. [12]. In *Gray Box Optimization*, the optimizer can evaluate the set of m subfunctions in Equation (1) (although their internal structure is unknown). This contrasts with Black Box Optimization, where the optimizer can only evaluate solutions and get their fitness value.

3.1 Variable Interaction Graph

The *Variable Interaction Graph* (VIG) [12] is a useful tool that can be constructed under Gray Box Optimization. It is a graph $VIG = (V, E)$, where V is the set of variables and E is the set of edges representing all pairs of variables (x_i, x_j) having *nonlinear interactions*. These nonlinear interactions can be captured in two ways. First, assuming that every pair of variables appearing together in a subfunction have a nonlinear interaction. A second approach is to apply the Fourier transform [9], and then look at every pair of variables to determine if there is a non-zero Fourier coefficient associated with a term with the two variables. This second method is more precise and not very expensive, because the Fourier transform can be constructed in $O(n)$ time for k-bounded epistasis functions.

An example of the construction of the variable interaction graph for a function with $n = 18$ variables (numbered from 0 to 17) and $k = 3$, is given below. We will refer to variables using numbers, e.g., $9 = x_9$. The objective function is the sum over the following 18 subfunctions:

$f_0(0, 6, 14)$	$f_5(5, 4, 2)$	$f_{10}(10, 2, 17)$	$f_{15}(15, 7, 13)$
$f_1(1, 0, 6)$	$f_6(6, 10, 13)$	$f_{11}(11, 16, 17)$	$f_{16}(16, 9, 11)$
$f_2(2, 1, 6)$	$f_7(7, 12, 15)$	$f_{12}(12, 10, 17)$	$f_{17}(17, 5, 2)$
$f_3(3, 7, 13)$	$f_8(8, 3, 6)$	$f_{13}(13, 12, 15)$	
$f_4(4, 1, 14)$	$f_9(9, 11, 14)$	$f_{14}(14, 4, 16)$	

From these subfunctions, assume we extract the nonlinear interactions that are shown in Figure 1. In this example, every pair of variables that appear together in a subfunction has a nonlinear interaction.

3.2 Efficient Local Search

For Mk landscapes, Whitley and Chen [11] proved that the location of improving moves can be determined in constant time for the Hamming distance 1 neighborhood. Two solutions are neighboring in this neighborhood if they differ in one bit. This result was later generalized by Chicano et al. [2], who proposed a hill climber that explores the solutions contained in a Hamming ball of radius r around a solution in constant time. The concept of *Score function* is at the core of both results. For $v, x \in \mathbb{B}^n$, and a pseudo-Boolean function $f : \mathbb{B}^n \to \mathbb{R}$, we denote the *Score* of x with respect to

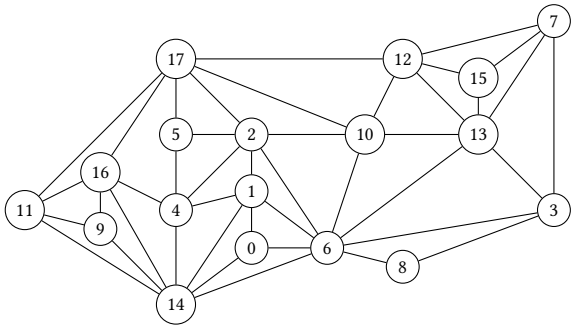


Figure 1: Sample Variable Interaction Graph (VIG).

move v as $S_v(x)$, defined as follows:

$$S_v(x) = f(x \oplus v) - f(x), \tag{2}$$

where $\oplus$ denotes the exclusive OR bitwise operation. The Score $S_v(x)$ is the change in the objective function when we move from solution x to solution $x \oplus v$, that is obtained by flipping in x all the bits that are 1 in v. Storing the Score function in memory makes it possible to explore all the solutions at Hamming distance 1 in constant time. When a move is performed, the Score function is updated. If the number of subfunctions a variable appears in is bounded by a constant, then the time required to update the Score function is constant, yielding a very efficient hill climber for pseudo-Boolean problems [2]. This hill climber is used in our implementation of Iterated Local Search used in the experimental evaluation.

4 METHODOLOGY

4.1 Benchmark Instances

We used benchmark instances having a low number of variables in order to be able to compute the global optimum in all of them. In particular, we used instances with 40 variables, which is the minimum number of variables found in instances of the MAX-SAT Evaluation 2017[1]. The ratio of the number of clauses to the number of variables is known to have an impact on the search difficulty [13]. Therefore, we generated random MAX-SAT instances with $n = 40$ variables and varying the number of clauses (m). In particular, we generated instances with $m/n \in \{2, 4, 6, 8, 10, 11\}$. The instance generator and the instances themselves can be found with the source code in GitHub[2].

We also considered three industrial instances from the MAX-SAT Evaluation 2017. The instances are `maxcut-san400_0.5_1`, `maxcut-sanr200_0.7` and `maxcut-brock400_2`, with $m = 790$, $m = 1092$, and $m = 1188$ clauses, respectively.

For all the instances we computed the global optimum using an exact method, in order to be certain that the sampling processes reached it. Instead of using an exhaustive enumeration (which could take a long time) we applied the recently proposed Dynastic Potential Crossover Operator (DPX) [1]. This operator is able to explore the full dynastic potential of two parent solutions providing the best among the solutions in that space. When one parent solution is exactly the complement of the second parent, the dynastic potential is the whole search space and the global optimum is provided. DPX worked well for the instances with the lowest number of clauses but failed to run for largest instances due to memory problems. To overcome this, we made equal some variables in both parent solutions, thus exploring only one hyperplane in the search space. After enumerating all the possible hyperplanes we can obtain the global optimum. The exploration of these hyperplanes was parallelized in different machines, thus reducing the time to compute the optimum from days to minutes, thanks to a cluster of more than 100 machines.

[1] http://mse17.cs.helsinki.fi/benchmarks.html
[2] https://github.com/jfrchicanog/EfficientHillClimbers

4.2 Sampling Method

The sampling procedure consists of aggregating the local maxima and transition edges obtained by 100 runs of an Iterated Local Search (Algorithm 1). The stopping condition was set as fixed running time (60 s). Weights are added to edges indicating the number of times they appear in the sampling process.

Algorithm 1 Iterated Local Search

```
1:  x ← generateRandomSolution();
2:  x ← applyLocalSearch(x);
3:  while not stopping condition do
4:      y ← perturb (x);
5:      y ← applyLocalSearch(y);
6:      reportEdge(x,y);
7:      if f(y) > f(x) then
8:          x ← y;
9:      end if
10: end while
11: return x;
```

Table 1: Description of Metrics.

Performance Metrics	
hitrate	Proportion of runs that reached the global optimum.
iter	Number of iterations before reaching the global optimum.
Network Metrics	
noptima	Number of optima (including local and global).
nglobal	Number of global optima.
$edges_i$	Proportion of edges that are improving.
$edges_n$	Proportion of edges that are neutral.
$edges_w$	Proportion of edges that are worsening.
ncoptima	Number of compressed optima (plateaus).
ncglobal	Number of compressed global optima.
ncedges	Number of compressed edges.
neutrality	Ratio of compressed to total number of optima.
lplateau	Size of the largest plateau.
nlfunnels	Number of sub-optimal funnels.

In our ILS implementation, the perturbation flips 5% of the variables selected at random (which corresponds to 2-bitflips for $n = 40$). The local search operator iterates the hill climber explained in Section 3.2 until a local optima is found (no neighbor can improve the objective function). A new local optimum is only accepted in Line 7 if it improves the incumbent solution. However, we report all the

edges encountered between local optima in Line 6, which includes neutral and worsening edges.

4.3 Network and Performance Metrics

For each instance, we extracted the LON models and computed the measurements described in Table 1. Metrics are reported as aggregations over 100 runs.

5 RESULTS

5.1 Network Analysis

Table 2 reports the network statistics described in Table 1 for the sampled local optima networks on the random and industrial instances with $n = 40$ and increasing clause to variable ratio m/n.

For the random instances, results show that the the size of the networks, as measured by the number of local optima, *noptima*, and compressed local optima, *ncoptima*, decreases with increasing number of clauses. This seems counter-intuitive, a small connected network should, in principle, be easier to traverse than a larger one, as the possible trajectories towards the global optimum are shorter. But we know that search difficulty tends to increase with the number of clauses [13]. Some network metrics help us to explain this. Looking at the number of global optima, *nglobal*, we can observe a sharp decline in this metric when going from $m/n = 4$ to 6. Moreover, looking at the size of the largest plateau size *lplateau*, we can observe that for $m/n \in \{1, 2\}$ the largest plateau is the global optimum plateau (as the size coincides with the number of global optima), but this is not the case for $m/n \geq 4$, where the largest plateau is a sub-optimal plateau. Another network metric explaining the increased difficulty for larger m/n ratios is the proportion of worsening edges, $edges_w$. This metric reflects the effort required across the search process to find an improving transition. This value is found to be larger for the instances with large m/n ratios. The instance with 440 clauses is the only random instance revealing a sub-optimal funnel, $nlfunnels = 1$. All the instances show a high level of neutrality, which is a well known feature of MAX-SAT fitness landscapes. This can be appreciated by looking at the proportion of neutral edges $edges_n$, and the ratio of compressed to total number of optima *neutrality*[3], which both decrease with the number of clauses.

Regarding the performance metrics in random instances, we observe that the hit rate is maximum in all the instances except the largest one. In these cases we can use the average number of iterations to reach the global optimum as a search difficulty metric and we observe an increase in this value with the number of clauses. The largest instance ($m/n = 11$) does not reach the global optimum in all the cases and this biased the average number of iterations to reach the global optimum, which cannot be compared with the other random instances. The conclusions is that the difficulty of the random instances increase with the number of clauses, as was observed in previous work [13]

The industrial instances show a larger proportion of worsening edges $edges_w$, when compared to the random instances, indicating higher search difficulty. The less constrained instance ($m/n = 20$) reveals a network of similar size than the studied random instances. However, the two instances with ratio $m/n > 1000$ show larger networks; they also show two global optima plateaus. The instance with largest ratio ($m/n = 1188$) shows a sub-optimal funnel. The industrial instances also show a high level of neutrality as indicated by the high *neutrality* value. Regarding the performance metric, we observe that the instance with $m = 1092$ clauses is the most difficult one. This highlights an interesting fact: in industrial instances more clauses do not necessarily means more difficulty. The number of improving edges seems to be a good indicator of search difficulty in this the industrial instances.

In order to give a more detailed characterisation of the plateau sizes, Figure 2 shows bar-plots of the sizes of compressed optima (plateaus) in decreasing order for all the studied instances. The instance type, number of clauses and clause to variable ratio are indicated in the sub-captions. In order to better compare the magnitude differences, the square root of the sizes is reported. The scale of the y axis goes from 0 to 25 for first three plots (a), (b) and (c), while for the rest, it goes from 0 to 10. A general trend across all instances is the existence of a few large plateaus, followed by a larger set of smaller plateaus, ending with a number of plateaus of size one (i.e. single optima). On the random instances (plots (a) - (f)), the size and number of plateaus decreases with increasing m/n ratios. In contrast, the number of plateaus does not seem to decrease with the m/n ratio on the industrial instances.

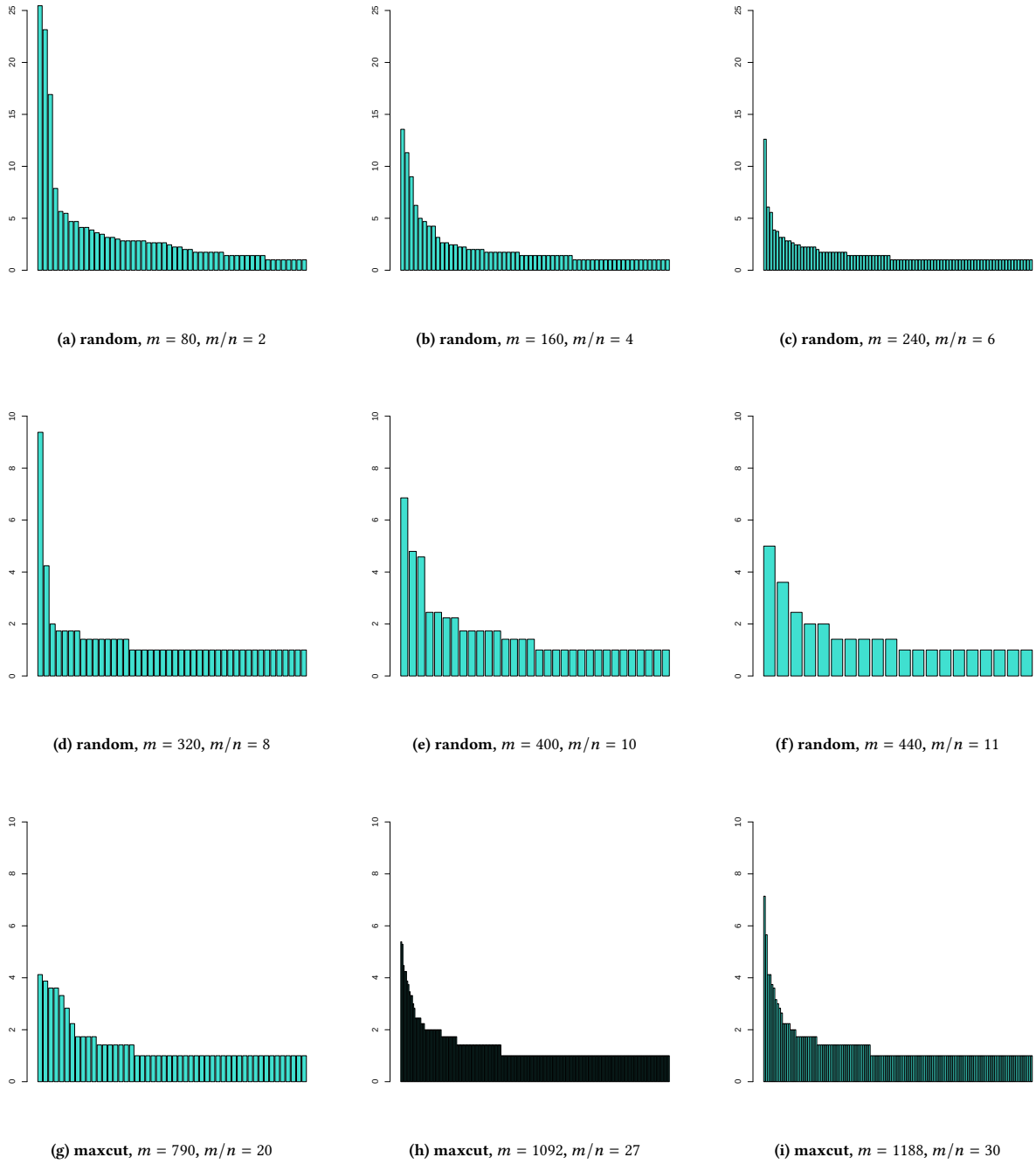
5.2 Visualization

Visualization is a useful tool in the analysis of network data, allowing us to appreciate structural features which could be difficult to infer from the raw data and statistical analysis. In particular, the sampled compressed monotonic LONs for the studied benchmark instances are relatively small and not very dense (with less than 300 nodes and edges as can be seen in Table 2), which facilitates visualization.

Figure 3 illustrates 2D projections of the LONs for all the studied instances. The instance type, number of clauses and clause to variable ratio are indicated in the sub-captions. Network plots were produced using the R statistical language together with the igraph package. Graph layouts consider *force-directed* methods. Networks are decorated to reflect features relevant to search dynamic. The rectangular nodes indicate plateaus with lengths proportional to plateau sizes (i.e. the number of single local optima within a plateau), while the circular nodes indicate single optima. The color of nodes indicates the funnel membership with pink reflecting nodes that belong to global optimal funnels, and light blue indicating nodes that belong to sub-optimal funnels. Red nodes correspond to the global optimum (optima), while dark blue nodes indicate the top of sub-optimal funnels. Edges widths are proportional to their weight, which is the estimated probability of transitions. That is, the most probable transitions are thicker in the plots.

On the random instances (plots (a)-(f) in Fig. 3), the overall size of the networks decreases with the m/n ratio. The amount of neutrality also decreases with the number of clauses, which can be appreciated in the network plots as the reduction in number and length of the rectangular nodes. For the instances with lower number of clauses (plots (a) and (b) in Fig. 3) the global optimum is a large plateau (red rectangle), whereas for $m/n > 4$ (plots (c) - (f) in

[3] *neutrality* is reported as as the reciprocal of the ratio of compressed to total number of optima, so that higher values represent higher neutrality.



(a) random, $m = 80$, $m/n = 2$ **(b) random**, $m = 160$, $m/n = 4$ **(c) random**, $m = 240$, $m/n = 6$

(d) random, $m = 320$, $m/n = 8$ **(e) random**, $m = 400$, $m/n = 10$ **(f) random**, $m = 440$, $m/n = 11$

(g) maxcut, $m = 790$, $m/n = 20$ **(h) maxcut**, $m = 1092$, $m/n = 27$ **(i) maxcut**, $m = 1188$, $m/n = 30$

Figure 2: Barplots indicating the distribution of local optima plateau sizes for the benchmark instances with $n = 40$. The instance type, number of clauses and clause to variable ratio are indicated in the sub-captions. The square roots of the plateau sizes is used to account for the magnitude differences across instances. Notice that the y axis of the first 3 plots (a), (b) and (c) goes from 0 to 25, while for the rest it goes from 0 to 10.

Table 2: Performance and network metrics for the random and industrial benchmark instances with $n = 40$ variables. The first two columns indicate the number of clauses m and clause to variable ratio m/n.

Random Instances													
m	$\frac{m}{n}$	*hitrate*	*iter*	*noptima*	*nglobal*	$edges_i$	$edges_n$	$edges_w$	*ncoptima*	*ncglobal*	*ncedges*	*neutrality*	*lplateau*	*nlfunnels*
80	2	1.00	27	9 678	648	0.003	0.705	0.292	52	1	89	0.995	648	0
160	4	1.00	20	1989	184	0.012	0.627	0.361	61	1	91	0.969	184	0
240	6	1.00	55	568	1	0.156	0.702	0.143	87	1	121	0.847	159	0
320	8	1.00	39	305	2	0.124	0.551	0.325	44	1	53	0.856	88	0
400	10	1.00	120	282	3	0.081	0.585	0.335	32	2	41	0.887	47	0
440	11	0.95	8	180	4	0.137	0.382	0.481	20	1	23	0.889	25	1
Industrial Instances (maxcut)														
790	20	1.00	93	437	5	0.125	0.263	0.611	50	1	76	0.886	17	0
1092	27	0.96	60424	3154	12	0.053	0.133	0.814	225	2	282	0.929	29	0
1188	30	0.99	408	1885	6	0.105	0.200	0.695	141	2	250	0.925	51	1

Fig. 3), the global optimum becomes a single node or a small plateau, with large sub-optimal plateaus appearing (pink rectangles). The random instance with $m = 440$, is the only one revealing a sub-optimal funnel Plot (b), which can be appreciated as the subset of blue nodes pointing towards the small dark blue node (the funnel top or sink).

The LON for less constrained of the industrial instances (plot (g) in Fig. 3) resembles in size and structure the LON of the random instance with $m/n = 8$ (plot (d)). However, the most constrained industrial instances (plots (h) and (i) in Fig. 3) show visibly larger networks, with two separated global optima small plateaus. Moreover, the industrial instance with $m/n = 30$ (plot (i)) also shows a very small sub-optimal funnel, visualised as the two light blue nodes, ending in a dark blue small plateau acting as trap to the search process (no outgoing edges).

6 CONCLUSIONS

We conducted a preliminary study extracting, analysing and visualising LONs for the MAX-SAT problem. This article joins two recent active research strands, Local Optima Networks and Gray-Box Optimization. The recently proposed Compressed Monotonic LON model allowed us to deal with the large plateaus observed in MAX-SAT, while Gray-box optimization allowed us a fast extraction of the LON data. We studied both randomly generated and industrial instances. All the instances studied showed high degrees of neutrality, as expected. On the random instances, increasing the number of clauses produced smaller networks. This seemed counter-intuitive initially, as smaller connected networks seem to reflect easier search. But we know that search difficulty in MAX-SAT tends to increase with the number of clauses. A closer analysis revealed that it takes longer for the algorithm to find improving transitions on the more constrained instances. So even though the monotonic trajectories towards the global optimum in the networks are shorter, it takes longer for the algorithm to find the improving hops. Moreover, the random instances with low number of clauses have a large global optima plateau, indicating that it is easier to reach it. The industrial instances showed a different pattern than the random instances, producing larger networks when the number of clauses increases. This difference deserves further investigation. We also observed the appearance of a sub-optimal funnel in the most constrained random instance and the most constrained industrial instance. Sub-optimal funnels are associated with increased search difficulty, another promising sign indicating that LON analysis can help in understanding search difficulty.

Future work will explore larger instances and will investigate correlations between LON features and algorithm performance. We will also explore LONs induced by hybrid algorithms incorporating partition crossover. The improving, neutral and worsening edges are biased by the stopping condition. The reason is that after the global optimum is found no improving edges can be found, thus reducing its proportion. To avoid this bias, in future work the analysis should be stopped once the global optimum is found.

ACKNOWLEDGEMENTS

This work is partially funded by the University of Stirling, the University of Malaga (Exhauro project) and the Ministry of Innovation and Competitiveness under contract (TIN2017-88213-R).

REFERENCES

[1] Francisco Chicano, Gabriela Ochoa, Darrell Whitley, and Renato Tinós. 2019. Quasi-Optimal Recombination Operator. In *Proceedings of the European Conference on Evolutionary computation for Combinatorial Optimisation.*

[2] Francisco Chicano, Darrell Whitley, and Andrew M. Sutton. 2014. Efficient identification of improving moves in a ball for pseudo-boolean problems. In *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014*, Dirk V. Arnold (Ed.). ACM, ACM, NY, USA, 437–444. https://doi.org/10.1145/2576768.2598304

[3] Fabio Daolio, Sébastien Vérel, Gabriela Ochoa, and Marco Tomassini. 2010. Local Optima Networks of the Quadratic Assignment Problem. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010*. IEEE, 1–8. https://doi.org/10.1109/CEC.2010.5586481

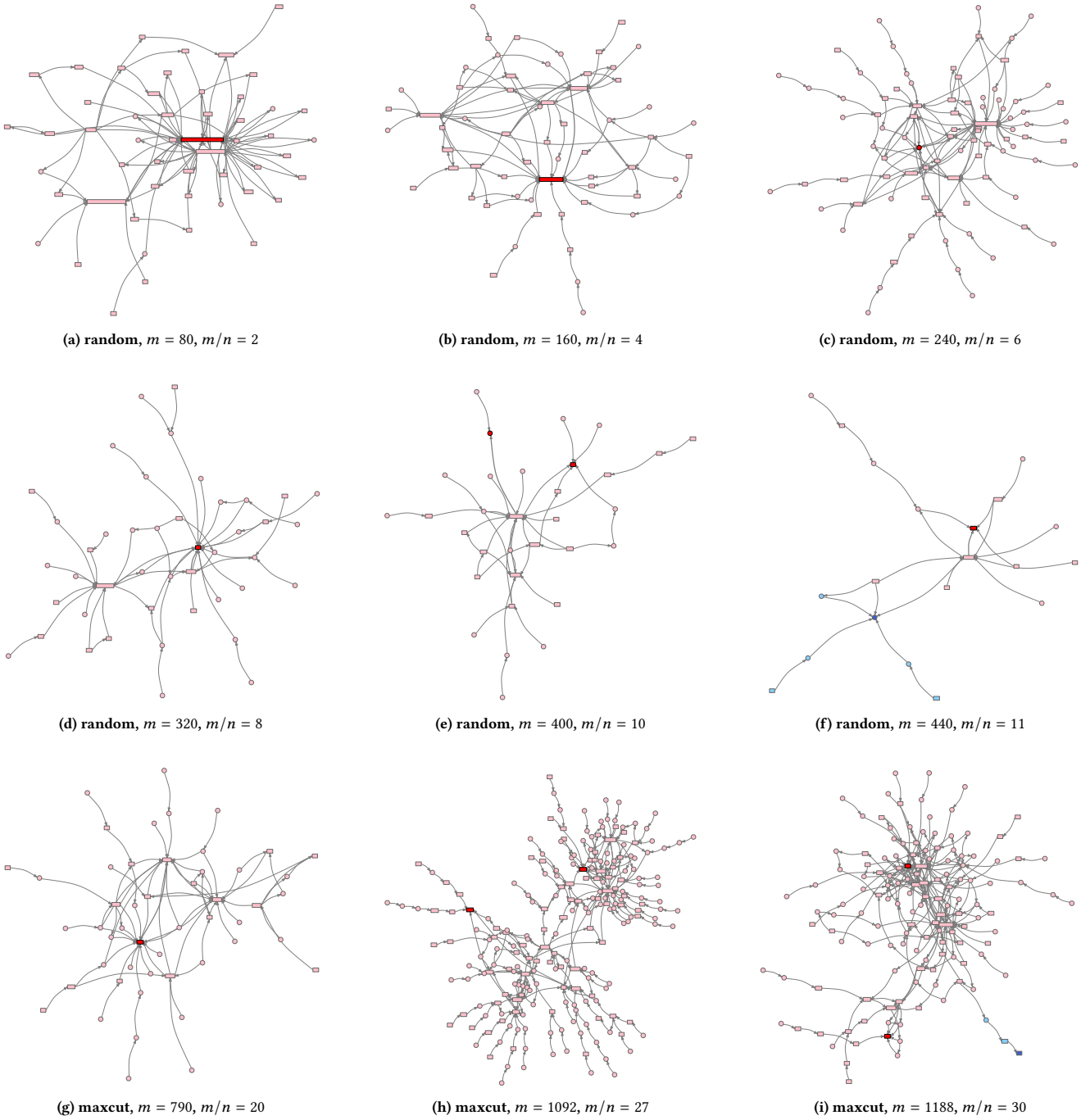
[4] Fabio Daolio, Sébastien Vérel, Gabriela Ochoa, and Marco Tomassini. 2013. Local Optima Networks of the Permutation Flow-Shop Problem. In *Artificial Evolution (Lecture Notes in Computer Science)*, Vol. 8752. Springer, 41–52. https://doi.org/10.1007/978-3-319-11683-9_4

[5] G. Ochoa, M. Tomassini, S. Verel, and C. Darabos. 2008. A Study of NK Landscapes' Basins and Local Optima Networks. In *Genetic and Evolutionary Computation Conference, GECCO*. ACM, 555–562.

[6] Gabriela Ochoa and Nadarajen Veerapen. 2016. Deconstructing the Big Valley Search Space Hypothesis. In *Evolutionary Computation in Combinatorial Optimization, EvoCOP 2016 (Lecture Notes in Computer Science)*, Vol. 9595. Springer, 58–73. https://doi.org/10.1007/978-3-319-30698-8_5

[7] Gabriela Ochoa, Nadarajen Veerapen, Fabio Daolio, and Marco Tomassini. 2017. Understanding Phase Transitions with Local Optima Networks: Number Partitioning as a Case Study. In *Evolutionary Computation in Combinatorial Optimization, (EVOCOP) (LNCS)*, Vol. 10197. Springer, 233–248.

[8] P. Stadler. 2002. Fitness landscapes. *Biological evolution and statistical physics* (2002), 183–204.



(a) random, $m = 80$, $m/n = 2$ **(b) random,** $m = 160$, $m/n = 4$ **(c) random,** $m = 240$, $m/n = 6$

(d) random, $m = 320$, $m/n = 8$ **(e) random,** $m = 400$, $m/n = 10$ **(f) random,** $m = 440$, $m/n = 11$

(g) maxcut, $m = 790$, $m/n = 20$ **(h) maxcut,** $m = 1092$, $m/n = 27$ **(i) maxcut,** $m = 1188$, $m/n = 30$

Figure 3: Local optima networks for the benchmark instances with $n = 40$ variables. The instance type, number of clauses and clause to variable ratio are indicated in the sub-captions. Rectangular nodes indicates plateaus, that is, nodes compressing two or more connected local optima, while circular nodes indicate single local optima. The lengths of rectangles is proportional to the size of plateaus, specifically, to the square root of the number of optima in the plateaus. Pink nodes belong to the funnel containing the global optimum, while light blue to sub-optimal funnels. Red indicates the global optimum(a), while dark blue indicates the bottom of a sub-optimal funnel.

[9] Audrey Terras. 1999. *Fourier Analysis on Finite Groups and Applications, Cambridge U. Press, Cambridge.* Cambridge University Press.

[10] S. Verel, G. Ochoa, and M. Tomassini. 2011. Local Optima Networks of NK Landscapes with Neutrality. *IEEE Transactions on Evolutionary Computation* 15, 6 (2011), 783–797.

[11] Darrell Whitley and Wenxiang Chen. 2012. Constant time steepest descent local search with lookahead for NK-landscapes and MAX-kSAT. In *Proceedings of GECCO*. ACM, NY, USA, 1357–1364. https://doi.org/10.1145/2330163.2330351

[12] Darrell Whitley, Francisco Chicano, and Brian W. Goldman. 2016. Gray Box Optimization for Mk Landscapes (NK Landscapes and MAX-kSAT). *Evolutionary Computation* 24 (Jan-09-2016 2016), 491 – 519. https://doi.org/10.1162/EVCO_a_00184

[13] Weixiong Zhang. 2001. Phase Transitions and Backbones of 3-SAT and Maximum 3-SAT. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming (CP '01)*. Springer-Verlag, Berlin, Heidelberg, 153–167. http://dl.acm.org/citation.cfm?id=647488.726965