UNIVERSITY OF SOUTHAMPTON

Analysis of the Fitness Landscape for the Class of Combinatorial Optimisation Problems

by

Mohammad- H. Tayarani- N.

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

May 2013

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Mohammad- H. Tayarani- N.

Anatomy of the fitness landscape for a group of well known combinatorial optimisation problems is studied in this research and the similarities and the differences between their landscapes are pointed out. In this research we target the analysis of the fitness landscape for MAX-SAT, Graph-Colouring, Travelling Salesman and Quadratic Assignment problems. Belonging to the class of NP-Hard problems, all these problems become exponentially harder as the problem size grows. We study a group of properties of the fitness landscape for these problems and show what properties are shared by different problems and what properties are different. The properties we investigate here include the time it takes for a local search algorithm to find a local optimum, the number of local and global optima, distance between local and global optima, expected cost of found optima, probability of reaching a global optimum and the cost of the best configuration in the search space. The relationship between these properties and the system size and other parameters of the problems are studied, and it is shown how these properties are shared or differ in different problems. We also study the long-range correlation within the search space, including the expected cost in the Hamming sphere around the local and global optima, the basin of attraction of the local and global optima and the probability of finding a local optimum as a function of its cost. We believe these information provide good insight for algorithm designers.

Contents

List of Figures

List of Tables

Nomenclature

$\mathscr{V}$	Vertex set of a Graph
$\mathscr{E}$	Edge set of a graph
p	Probability of edges in a graph
n	Size of a problem
k	Number of colours in a graph colouring problem
α	Number of clauses to variables ratio in Max-Sat problem
β	Number of literals in each clause in the Max-Sat problem
$c(.,.)$	Cost function
$f(.,.)$	Fitness function
$D(.,.)$	Distance between two configurations
h	Hamming distance
g_i	i-th clause in a problem instance of Max-Sat problem
$\pi(.)$	Permutation operator
$\boldsymbol{x}$	Configuration in the landscape
$\mathscr{N}(\boldsymbol{x})$	Neighbourhood set of the configuration $\boldsymbol{x}$
$\mathscr{X}$	Set of configurations
$\mathscr{L}$	Set of local optima
$\mathscr{F}$	Set of facilities in Quadratic Assignment Problem
$\mathscr{Q}$	Set of locations in Quadratic Assignment Problem
$\mathbf{M}$	Problem matrix in Travelling Salesman Problems.
$P_v(c)$	Probability of visiting a local optimum at cost c
$N(c)$	Number of local optima at cost c.
μ_i	i-th moment of a distribution
$\chi(G)$	Chromatic number of a graph

Acknowledgements

I would like to thank my supervisor, Dr Adam Prügel Bennett for his constructive comments.

To my wife, who was always there for me. Also to my little daughter, Diana, who is the colour of my life.

Chapter 1

Introduction

To design successful heuristic algorithms for hard combinatorial problems requires an understanding of the fitness landscape structure. This knowledge about the fitness landscape, provides good insights into problems for algorithm developers. This will be problem and even instance dependent. Nevertheless, in this research we will show that many of the landscape properties are common across many instances and even across different problem classes.

The aim of this research is to study the fitness landscape properties of a group of combinatorial optimisation problems, including the Graph-Colouring, the MAX-SAT, the Travelling Salesman and the Quadratic Assignment. These problems, although abstracted from the real world, are chosen to be representative of the types that practitioners face. We try to identify those features which are common to a large class of optimisation problems, and those features that critically distinguish the problems. We focus mainly on long-range properties which we study by using the result of many local searches to find local optima for graphs with up to 100 vertices. We attempt to extrapolate from these relatively small instances to understand the behaviour of large problems. We regard the long-range landscape properties as a key to the success of evolutionary algorithms. However, for the sake of completeness we also study other properties such as the density of configurations and the auto-correlation, which are frequently studied in papers on landscape analysis.

In this thesis we have studied a large number of properties for all these problems. However, each section is relatively self-contained, so we hope that reading it does not prove too arduous. None of the properties we analyse is, in itself, that surprising, but we believe that the picture of the landscape which is revealed is informative for researchers interested in developing algorithms for solving combinatorial optimisation problems. By pointing out the similarities and the dissimilarities in some important properties of the fitness landscape, we show how some features change from a problem to another, and how some features are common among a group of optimisation problems. This may provide good information about the fitness landscape of the combinatorial optimisation problems; the property that should be of interest to readers wishing to develop more general purpose algorithms.

1.1 Literature Survey

The concept of fitness landscape, introduced by Wright (1932) to demonstrate the dynamics of biological evolutionary optimisation, has been useful for the analysis and understanding of the behaviour of evolutionary algorithms. Fitness landscape analysis techniques are used to better understand the influence of genetic representations and associated variation operators when solving a combinatorial optimisation problem (McCarthy, 2008; Tavares et al., 2006, 2008; Riley and Ciesielski, 2010). This understanding can provide useful information about the structure of the problem and the type of operators that are better for particular problems (Merz and Freisleben, 1998a; Newth and Brede, 2006; Slany and Sekanina, 2007; Czogalla and Fink, 2009). Furthermore, the study of fitness landscape can be useful in designing evolutionary algorithms or hybrid algorithms (Moscato, 1989; Moscato and Norman, 1992; Qasem and Prügel-Bennett, 2008), since the landscape analysis can help us to predict the performance of the proposed algorithms (Shaowei and Qiuping, 2007; Huanga et al., 2009). Some researchers use the landscape analysis to study some parameters of the evolutionary algorithms, like the population size (Alander, 1999), or some operators like mutation and crossover (Mathias and Whitley, 1992a; Suzuki and Iwasa, 1997), the recombination operators (Hornby, 1996), or the perturbation operator (Martin et al., 1999). Some researchers have used the landscape analysis to explain why some algorithms, like local search algorithms (Fonlupt et al., 1997), memetic algorithms (Merz, 2004) or metaheuristic algorithms based on local search (Watson, 2010), work better on particular landscapes. Such studies will clearly be problem and even instance dependent; nevertheless some statistical properties are common across many instances and even across different problem classes.

There is a number of works trying to exploit the concept of the landscape and landscape analysis in developing new sets of algorithms for different problems. The fitness landscape analysis is used to propose memetic algorithms for graph bi-partitioning problem (Merz and Freisleben, 1998b), Resource Allocation problem (Huang et al., 2009) and Maximum Satisfiability problem (Zhang et al., 2003; Zhang, 2004), or improving the performance of evolutionary algorithms by a landscape approximation (Ratle, 1998; Pošík and Franc, 2007; Shen and He, 2010). In a more recent work, the landscape analysis is used to propose a new population based algorithm (Qasem and Prügel-Bennett, 2010).

Examples of the measures that have attempted to capture the ruggedness of the landscape include the auto-correlation (Weinberger, 1990; Angel and Zissimopoulos, 1998; Czogalla, 2008) and fitness distance correlation (Jones, 1995a; Merz and Freisleben, 2000; Lefticaru and Ipate, 2008; Manderick et al., 1991). However it was soon realised that these measures do not necessarily capture the hardness of the problems, when a counter example was proposed which was an easy problem but showing no relationship between the fitness of the solutions and their distance to the global optimum (Altenberg, 1997). The question of "what makes a fitness landscape hard" continues to (Forrest and Mitchell, 1993; Jones, 1995a), and new methods of capturing the hardness of the problems are proposed, like algebraic properties of the solutions in the landscape (Grover, 1992; Stadler, 1995), modality or number of local optima (Horn and Goldberg, 1995) and the fractal dimension of the fitness landscape (Hoshino et al., 1998). Some researchers try to explain when a problem becomes hard, by studying the area in the landscape called "Olympus", in which the better local optima are located (Vérel et al., 2007, 2008). Fitness cloud is another method proposed to visualise the fitness landscape which tries to represent some properties of the fitness landscape (Collard et al., 2007; Vanneschi et al., 2007; Lu et al., 2011), reflecting the problem hardness.

During the last two decades many researchers have studied the landscape of optimisation problems including Travelling Salesman (Mathias and Whitley, 1992b; Stadler and Schnabl, 1992; Boese, 1995), Quadratic Assignment (Merz and Freisleben, 2000), Knapsack (Yoshizawa and Hashimoto, 2000; Tavares et al., 2008), MAX-SAT (Weixiong and Zhang, 2004; Qasem and Prügel-Bennett, 2010; Prügel-Bennett and Tayarani-N., 2011), graph drawing (Lehn and Kuntz, 2001), Graph-Colouring (Hertz et al., 1994b,a; Hamiez and Hao, 2001; Culberson and Gent, 2001; Bouziri et al., 2009, 2011), evolutionary antenna design (Alander et al., 2002), flow-shop scheduling (Czogalla and Fink, 2011) and Bayesian network structure (Wu et al., 2011) problems.

The previous research have mainly focused on studying some elementary properties of the fitness landscape, like auto-correlation or fitness distance correlation which are not very informative. For example auto-correlation measure gives some knowledge about the landscape ruggedness, which does not necessarily show the problem hardness or does not provide good information for algorithm designers. In this respect there is a need to studying new set of properties which are more representative of the problems nature and difficulties and are more useful for heuristic algorithm designers. This encouraged us to study new properties which have not been looked at before and are more informative. One example of these properties is the time it takes for a local search algorithm to find the local optima and the size of plateaux on which a local search algorithm wonders in search of a local optimum. In terms of problem difficulty, these properties are more informative than the elementary properties like auto-correlation as they describe how time consuming it is for a local search algorithm to find a local optimum. From the point of view of algorithm designers, the existence of plateaux could be very informative, as instead of just randomly wondering on the plateaux, a search algorithm, like Tabu search, which tries not to take repetitive steps could be more successful.

This research is different from the previous works in that we, for the first time, are studying a wide range of properties like time to local optima or the number of local optima and their relationship with the size or other parameters of the problems for a group of problems. This is, to the best of our knowledge, the first major attempt to study these properties for a group of completely different problems. These problems, by definition, are different and yet in some properties they show very similar behaviours. These similarities and the differences could provide good insight for evolutionary algorithm designers.

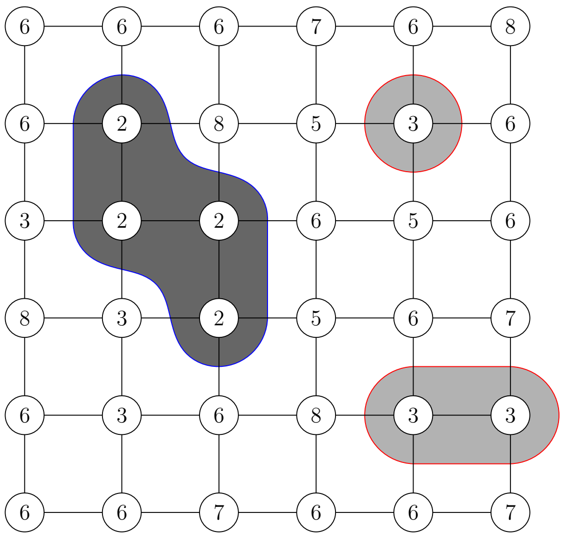


FIGURE 1.1: Schematic of the search space where each node represents a configuration. The edges show the neighbours connected to each configuration. The cost of each configuration is shown by the number. In this illustration there is one global minimum of cost 2 consisting of 4 configurations, and two local minima of cost 3 consisting of 1 and 2 configurations. Of course, the search space has a very different topology.

This thesis is a combination of five journal papers. For each of the problems studied in this research, namely the MAX-SAT (Prügel-Bennett and Tayarani-N., 2011) (published), Graph-Colouring (Tayarani-N and Prügel Bennett, 2011) (submitted), Travelling Salesman (Tayarani-N and Prügel Bennett, 2012c) (submitted), and Quadratic Assignment problems (Tayarani-N and Prügel Bennett, 2012b) (submitted), we have written one journal paper. Each paper now is a chapter in this thesis. The last chapter of this thesis, the comparison chapter, is another journal paper which studies the similarities and dissimilarities between the landscape of different problems (Tayarani-N and Prügel Bennett, 2012a) (accepted).

1.2 Methodology

In this research we study the landscape of the problems with respect a neighbourhood. This depends on the problems. For example for the MAX-SAT and the Graph-Colouring problems we use the Hamming neighbourhood. That is, we consider two configurations to be neighbours if they have a different value at one and only one variable. However, note that the neighbourhood definition may be slightly different for different problems, about which we will talk in the corresponding sections. The major analysis we undertake is to study the structure of the local minima with respect to cost. Note that we define a local minimum to be a connected set of configurations with no lower-cost neighbours. The local minimum (or minima) with the lowest cost is the global minimum (or minima). Our notion of what we mean by a local and global minima is illustrated schematically in Figure 1.1. Note that in our definition each local minimum can consist of several configurations.

Note however that the properties and the analyses provided in this thesis depend on the distance measure and the neighbourhood structure used and may not be true for other neighbourhood structures if significantly different from the ones used in this paper. Nevertheless we believe that the Hamming neighbourhood is the simplest and most widely used distance measure and so the analyses are useful for a great number of algorithm designers.

One way to study the topology of a fitness landscape is to search for local optima by an exhaustive search of the landscape; however, this would confine us to studying small problems with sizes up to 20 or 30 (see, for example, (Hallam and Prügel-Bennett, 2005) and (Benfold et al., 2007)). Instead we use a local search algorithm, which allows us to study larger instances. We run the local-search algorithm multiple times to find many local optima. To know if we have reached a local optimum we exhaustively search the set of configurations at the current cost to see if there are any lower-cost neighbours. If there is a lower-cost neighbour we move to it; otherwise, we have explored all the neighbours at the current cost so we can be sure that we have reached a local minimum. We can accomplish this cheaply by maintaining two sets. A set of configurations, $\mathscr{X}$, at the current cost, and a set of configurations, $\mathscr{U} \subset \mathscr{X}$ whose neighbours we have to check. The details of the exhaustive hill-climbing algorithm are given in Algorithm 1.

Algorithm 1 Exhaustive Hill Climbing Algorithm

```
ExhaustiveHill Climbing Algorithm
1.  $\boldsymbol{x} \leftarrow \texttt{randomConfiguration}()$.
2.  do
3.    $\mathscr{X} \leftarrow \{\boldsymbol{x}\}$
3.    $\mathscr{U} \leftarrow \{\boldsymbol{x}\}$
4.    $c \leftarrow \texttt{cost}(\boldsymbol{x})$
4.    while $(|\mathscr{U}| > 0)$
5.      $\boldsymbol{y} \leftarrow \texttt{chooseMember}(\mathscr{U})$
6.      $\mathscr{U} \leftarrow \mathscr{U} / \{y\}$
7.      forall $\boldsymbol{z} \in \texttt{Neighbours}(\boldsymbol{y})$
8.        if $(\boldsymbol{z} \notin \mathscr{X})$
9.          if $(\texttt{cost}(\boldsymbol{z}) < c)$
10.           $\mathscr{X} \leftarrow \{\boldsymbol{z}\}$
11.           $\mathscr{U} \leftarrow \{\boldsymbol{z}\}$
12.           $c \leftarrow \texttt{cost}(\boldsymbol{z})$
13.         if $(\texttt{cost}(\boldsymbol{z}) = c)$
14.           $\mathscr{X} \leftarrow \mathscr{X} \cup \{\boldsymbol{z}\}$
15.           $\mathscr{U} \leftarrow \mathscr{U} \cup \{\boldsymbol{z}\}$
16. return $(c, \mathscr{X})$
```

In algorithm 1 the function `randomConfiguration`$()$ returns a random configuration, `cost`$(\boldsymbol{x})$ returns the cost of a configuration $\boldsymbol{x}$, `chooseMember`$(\mathscr{U})$ chooses a member of the set $\mathscr{U}$, and `Neighbours`$(\boldsymbol{y})$ returns the set of neighbours of configuration $\boldsymbol{y}$. The set $\mathscr{U}$ can be efficiently implemented as either a stack or a queue. By careful book-keeping we can efficiently compute the cost of each neighbour and maintain a list of neighbours at the same or lower cost than the current cost. Note that the algorithm returns the cost of the local optimum and the set of configurations in the local minimum.

The algorithm is called exhaustive, because it explores the plateaux all the connected configurations at the same cost to make sure a local optimum is reached.

Chapter 2

MAX-SAT

In this chapter the fitness landscape of the MAX-SAT problem is studied. First we define the problem and the local search algorithm, then we study different properties of the fitness landscape.

2.1 Problem Definition

In this section, we describe the MAX-SAT problem and then specify the set of instances that we shall consider. We finish the section with a discussion of GSAT, the classic local-search algorithm for MAX-SAT.

The MAX-SAT problem is closely related to the satisfiability decision problem colloquially known as SAT. This problem involves a set of Boolean variables $\boldsymbol{x} = (X_1, X_2, \ldots, X_n)$ and a set of disjunctive clauses consisting of a subset of literals (a literal is either a variable or its negation). For example, a clause might be $X_1 \vee \neg X_5 \vee X_{10}$. Each clause can be considered an additional constraint that must be satisfied. In SAT the question is: “does there exist an assignment of the variables which satisfies all the clauses?”. Stephen Cook famously showed that any non-deterministic Turing machine can be reduced to a SAT instance whose size is a polynomial of the tape length, thus establishing that SAT is NP-complete (Cook, 1971). A special variant of SAT is β-SAT which consists of clauses containing exactly β literals. There is a staightforward polynomial reduction of any SAT instance to a 3-SAT instance. Thus, 3-SAT is NP-complete while 2-SAT can be solved in polynomial time.

MAX-SAT is the generalisation of SAT to problems which are not fully satisfiable. It asks the question whether there exists an assignment of the variables which satisfies all but T clauses. MAX-β-SAT is NP-hard for $\beta \geq 2$ (thus MAX-2-SAT is NP-hard even though 2-SAT is not (Garey and Johnson, 1979)). We will treat MAX-SAT as an optimisation problem and in particular we consider MAX-3-SAT. We use as the objective function the number of satisfied clauses, which we seek to maximise. Assuming there are m clauses and denoting the clauses by $g_i(\boldsymbol{x})$, then the fitness is given by

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} [\![g_i(\boldsymbol{x}) \text{ is satisfied}]\!]$$

where $[\![g_i(\boldsymbol{x}) \text{ is satisfied}]\!]$ is an indicator function equal to 1 if clause i is satisfied and 0 otherwise (i.e., all the literals in clause i are false).

Our focus will be on randomly generated instances, where each clause consists of β randomly chosen variables which are negated with probability of a 0.5. We require each variable in a clause to be different and all clauses to be unique. We denote the number of variables by n, the number of clauses by m, and the ratio of clauses to variables by α. As mentioned in the introduction, the ensemble of problem instances undergoes a phase transition for large n, such that almost all instances are satisfiable below a critical ratio of clauses to variables of α_c and unsatisfiable above this critical value. The value of α_c depends on β; for $\beta = 3$ the critical value is $\alpha_c \approx 4.3$, while for $\beta = 4$ the critical value is $\alpha_c \approx 9.8$. One reason for concentrating on $\beta = 3$ rather than $\beta > 3$ is that, for larger β, we must use considerably larger ratios of clauses-to-variable to find hard instances than is necessary for MAX-3-SAT. This would slow down the search algorithm preventing us from collecting as much data as we could for $\beta = 3$.

2.2 GSAT

This research focuses on heuristic search algorithms for MAX-SAT. As a baseline algorithm, we consider the classic GSAT local-search algorithm (Selman et al., 1992). GSAT is a hill-climbing algorithm which at each step chooses to change the variable that gives the best fitness improvement. When there are multiple alternatives it chooses to change one of them selected uniformly at random. Superficially, GSAT may appear inefficient as it considers all variables at each step, however, through judicious book-keeping GSAT can be made so that a single step requires $O(\alpha\beta^2)$ computations. It is therefore extremely fast (notice that it does not scale with n) and, in consequence, is very difficult to beat. Appendix A describes the implementation details of GSAT including a rather subtle set data structure which sped up our implementation over any other implementation that we are aware of.

As well as using GSAT as a baseline algorithm we also used it to find local optima as part of our empirical investigation of the fitness landscape. This, however, requires a method to determine whether a local optimum has been reached. To determine whether GSAT reaches a local optimum we switch to the exhaustive search algorithm described in section 1.2, i.e., algorithm 1.

One of the limiting constraints in carrying out our analysis is the memory requirements of exhaustive search. This becomes prohibitive for large n (above a few hundred at $\alpha = 8$) and close to the phase transition when the constant-fitness plateaus become very large.

We define a *plateau* as a large connected set of configurations at the same cost with a restricted exit to lower cost (fitter) configurations. The search algorithm has to explore the configurations in the plateau to find the exit. This can significantly slow down search if the plateau is large. The conditions for the existence of a plateau are quite severe. Not only must there be many configurations at the same cost, but they must be connected and only a small proportion of them can have neighbours at lower cost.

In other words, a solution with hundreds of neighbours at the same cost may be a cliff, if it has just one neighbour at lower cost (a better neighbour). However, having similar values in the problem matrices is not a sufficient condition for the existence of a plateau and greater time to local optima. In order for the solutions to make plateaus they have to have equal costs, be neighbours, and be surrounded by the solutions at the same or higher costs (worse solutions). This has an important consequence. For the problems with high similarity proportion, but small number of steps on the plateaus, there may be many neighbour solutions with equal cost, or there may even be large connected regions of equal cost solutions in the landscape, but they may not form a plateau. This is because the plateaus are the connected regions with no better neighbour.

2.3 Landscape Analysis

In this section, we look at some of the properties of the fitness landscape for MAX-SAT and in particular we concentrate on the scaling behaviour of different quantities as the system size grows. This analysis reveals why problem instances become difficult for local-search algorithms as they become large.

2.3.1 Density of States

We start by considering the number of configurations at each fitness level. The mean fitness for any Max-β-Sat problem is $f_{av} = (1-2^{-\beta})\alpha n$. We can compute the spread of fitnesses around the mean through random sampling (this will miss rarely occurring fitness values). Figure 2.1 shows the logarithm of the histogram of fitnesses around the mean fitness scaled by $\sqrt{\alpha n}$ for single instances of size 100, 1000 and 10 000. The results are almost identical for all randomly drawn instances (data not shown). The curves in Figure 2.1 are approximately quadratic indicating that the distributions are approximately normally distributed around their mean. The variance is empirically found to be around $0.1\alpha n$. This picture remains true for different values of α as shown in Figure 2.2.

We can understand the behaviour of the density of states by assuming that the clauses are independent of each other. In this case, the fitness is just the sum of αn independent Boolean random variables, with a probability of $1-2^{-\beta}$ of being 1 (i.e., the clause is satisfied). By the central limit theorem, we would expect the distribution of fitnesses to be approximately normally distributed with mean $(1-2^{-\beta})\alpha n$ and variance $2^{-\beta}(1-2^{-\beta})\alpha n$. This is close to the

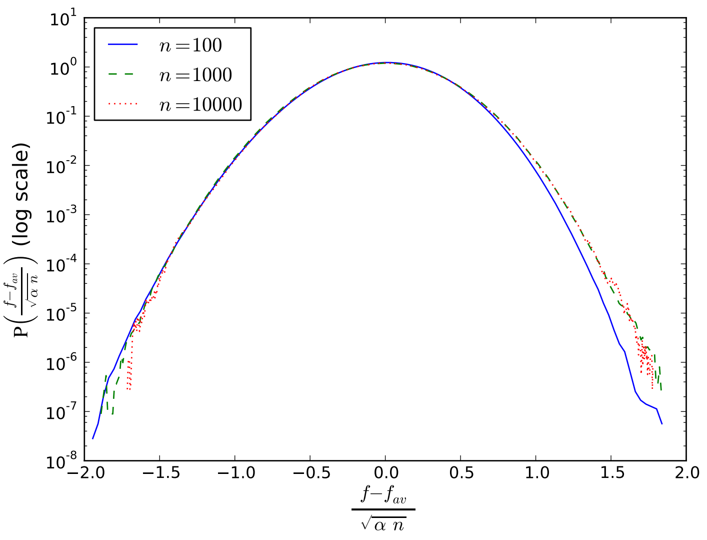


FIGURE 2.1: Logarithm of histogram of fitnesses computed by sampling 10^9 random configurations for single instances with $\alpha = 8$ at $n = 100$, 1000 and 10 000.

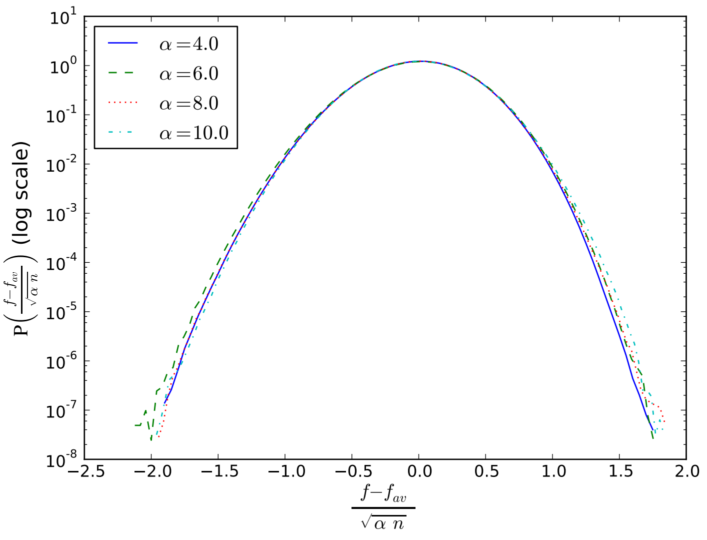


FIGURE 2.2: Logarithm of histogram of fitnesses computed by sampling 10^9 random configurations for single instances with $n = 100$ at $\alpha = 4$, 6, 8 and 10.

observed behaviour around the mean fitness. For any particular instance, some of the clauses will share the same variables, and so their truth values are correlated. As a consequence, the distribution of fitnesses will deviate from a normal distribution, particularly close to its tails. Approximately normal behaviour of the density of states is observed in other models where the objective function consists of a number of approximately independent components. For example, this is true in a large number of constraint satisfaction problems where the objective function counts the number of violated constraints. This includes both easy problems such as onesmax and hard problems such as graph-colouring.

2.3.2 Auto-correlation

We can measure the auto-correlation of a random walk through the search space, which is often used as a measure of the ruggedness of the fitness landscape (Weinberger, 1990). To compute this, we consider a random walk starting at an arbitrarily chosen initial configuration and moving to a Hamming neighbour at each step. Let $f(t)$ be the fitness at step t, then the auto-correlation is given by

$$\mathscr{R}(\tau) = \frac{1}{\sigma^2}\mathbb{E}\left((f(t+\tau) - f_{av})(f(t) - f_{av})\right),$$

where σ^2 is the variance in the fitness for random configurations. We have computed the auto-correlation for the same three instances used in Figure 2.1. These are shown in Figure 2.3. We see that, under this scaling, the auto-correlations are remarkably similar. For large τ the auto-correlation function appears to drop off approximately exponentially as

$$\mathscr{R}(\tau) \sim \varepsilon^{-\tau/l},$$

where l is known as the correlation length (Stadler, 1996). Empirically $l \approx 0.4n$. The correlation length is taken to be a measure of the landscape ruggedness—the smaller l the more rugged the landscape. We observe that, as n increases, the ruggedness decreases. That is, if we consider two instances of size n and n' we get roughly the same level of ruggedness if we make n'/n random steps on the instance of size n' as we would for a single step on the instance of size n.

Interestingly, the correlation length does not alter significantly with α. This is shown in Figure 2.4 (to emphasise the approximate exponential fall off we have plotted the logarithm of $\mathscr{R}(t)$). The auto-correlation length suggests that the landscape is relatively smooth with long-range correlations. We will see the origin of the long-range correlation and evaluate one of its properties analytically in section 2.4.

2.3.3 Time to Local Optimum

In much of the analysis of MAX-3-SAT, we will study properties of the local and global optima. We begin this analysis by considering the time taken by GSAT to reach a local optimum. To

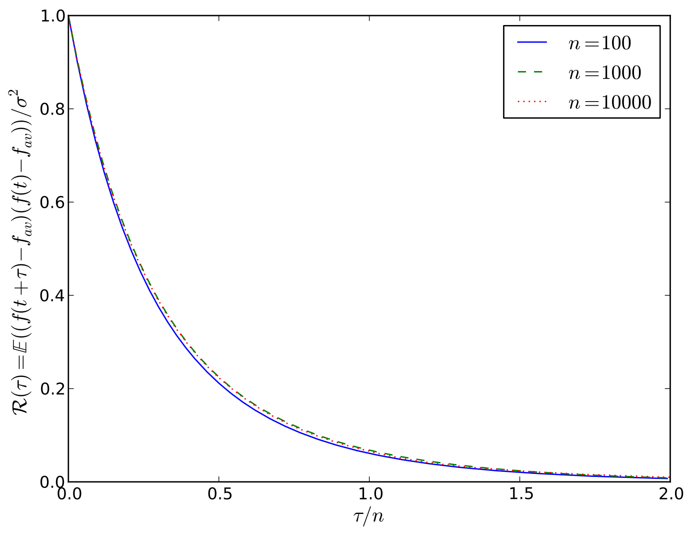


FIGURE 2.3: Auto-correlation for three instances of size $n = 100$, 1000, 10 000 with $\alpha = 8$ plotted against the time difference τ/n.

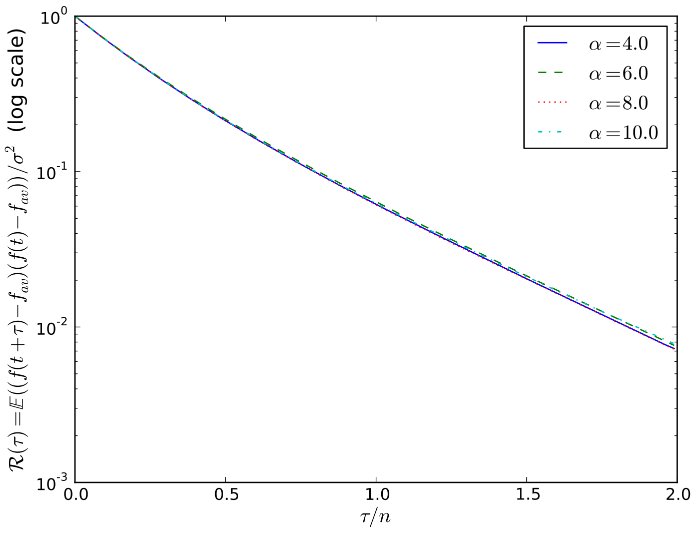


FIGURE 2.4: Logarithm of the auto-correlation for four instances of size $n = 100$ and $\alpha = 4$, 6, 8, 10 plotted against the time difference τ/n.

check if a local optimum has been reached we use the exhaustive search algorithm described above, however, if we find that we have not reached a local optimum we carry on GSAT from where we left off. The time taken to reach a local optimum depends critically on α. For $\alpha < \alpha_c$ the time to reach a local (and usually the global) optimum is relatively short. However, for $\alpha \approx \alpha_c$ the time to reach a local optimum increases rapidly. This is illustrated in Figure 2.5, where we show the mean and median times for GSAT to reach a local optimum plotted against α for instances of size $n = 50$. The large increase around $\alpha = 4$ is indicative of large plateau regions in the solution space. We could compute this only for small instances as the computation time, and the memory requirement to check if we have reached a local optimum becomes prohibitive around the phase transition (i.e., for $\alpha \approx \alpha_c = 4.3$). Interestingly, the big jump in the mean time taken to reach a local maximum around the phase transition is not reflected in the auto-correlation function, showing that the auto-correlation function is a comparatively poor indicator of the performance of search algorithms.

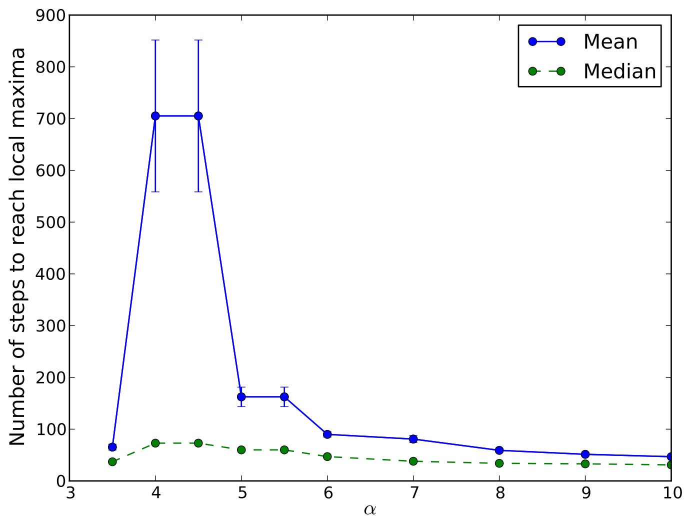


FIGURE 2.5: Mean time to reach a local optimum versus the ratio of clauses to variables, α. Each data point represents the mean over 10 instances and 100 hill-climbs per instance.

This section will concentrate on the regime $\alpha > \alpha_c$. For the sake of consistency we give results for $\alpha = 8$, although the same qualitative behaviour is observed at other values of α in this regime. As observed in Figure 2.5, the mean is considerably higher than the median, indicating that the distribution of times to reach a local maximum has a long tail. Even away from the phase transition, the number of steps to reach a local optimum can vary considerably. Figure 2.6 shows this distribution plotted on a semi-log scale to emphasise the rare events. This data was gathered on a single randomly-chosen problem instance. We notice that, on rare hill-climbs, it can take a exceedingly long time to hit a local optimum even at $\alpha = 8$.

Figure 2.7 shows the mean time to hit a local optimum versus the problem size, n. The graph shows that the time increases super-linearly, but still polynomially. The super-linear increase is due to the growth in the size of the plateau regions. Plotting the same data on a log-log scale,

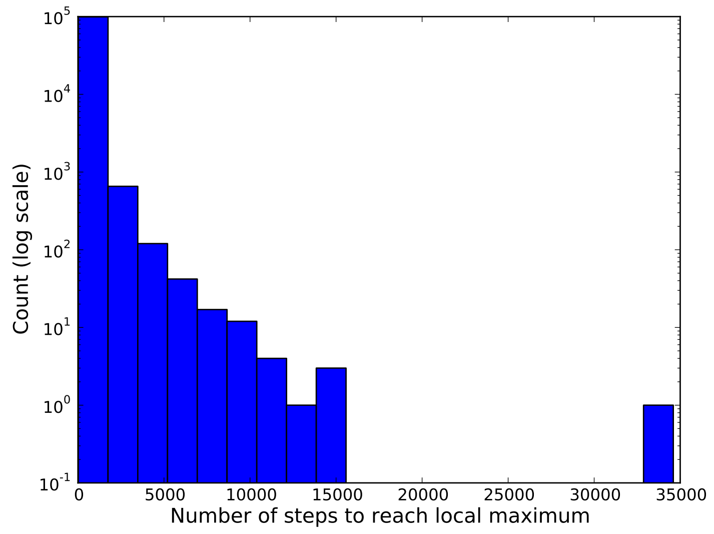


FIGURE 2.6: Count of occurrences that number of steps to reach a local maximum occurs in each bin plotted on a logarithmic scale. The data is for a single instance with $n = 100$ and $\alpha = 8$ collected on 100 000 hill-climbs.

Figure 2.8 indicates that the time to reach a local optimum appears to increase sub-quadratically. The graph shows the best straight line fit to the data. Using this extrapolation, the mean time to reach a maximum for a problem of size 10 000 would be 660 000 steps, while the median time is 320 000. Although some caution is needed in extrapolating from small instances to large instance, nevertheless, for $\alpha = 8$ it is fairly clear that reaching a local optimum is not hugely demanding.

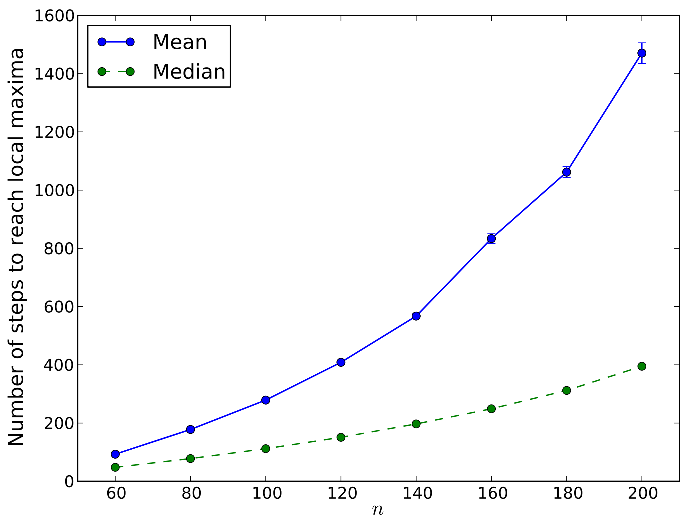


FIGURE 2.7: Mean time for GSAT to reach a local optimum versus the problem size n for fixed $\alpha = 8$. Each data point represents the mean over 100 instances and 1000 hill-climbs per instance.

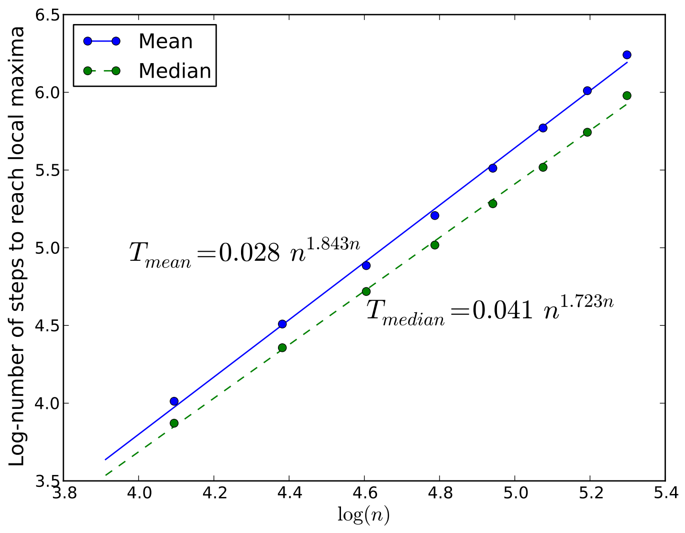


FIGURE 2.8: Natural logarithm of the mean time to reach a local optimum versus the natural logarithm of the problem size. This shows the same data as in Figure 2.7, but using a log-log axis. In addition, the best straight line fit to the data is shown.

2.3.4 Number of Local Optima

What makes large MAX-SAT instances difficult in this regime (i.e., $\alpha > \alpha_c$) is the large number of local optima. To investigate this, we ran GSAT until there was no improvement in 100 steps. To check whether we reached a local optimum, the exhaustive search algorithm was run. If a fitter configuration is found then the exhaustive search is re-initialised from this fitter solution. This is repeated until a local optimum is reached. The full search is repeated from a large number of randomly chosen starting configurations. Each local maximum is recorded together with the number of times that it is hit. Of course, we have no guarantee that we have reached every local maximum, particularly if a local maximum has a small basin of attraction. Typically there are some local maxima which have very small probabilities of being visited. For example, in one typical instance with $n = 50$, after 10^7 hill-climbs, there were 375 local optima found of which 5 were discovered less than 10 times. For another randomly chosen instance this time with $n = 100$, again after 10^7 hill-climbs, there were 25 126 local optima found of which 13 764 where found less than 10 times and 5150 local optima which were visited only once. However, those local optima that have been observed only once are all low fitness local optima (fitness less than or equal to 774, where the maximum fitness is 782).

Figure 2.9 shows the mean and minimum number of times each local optimum was found in 10^7 attempts. Of course, there are likely to be local optima that we have not found. However, note that, for fitnesses greater than 776, we have visited each local optimum at least 100 times. Thus with high probability any local optimum with a fitness greater than 776 that we have not found, would have a basin of attraction 100 times smaller than those that we have found. We call high fitness optima with abnormally small basins of attraction *elves*. Although we cannot rule

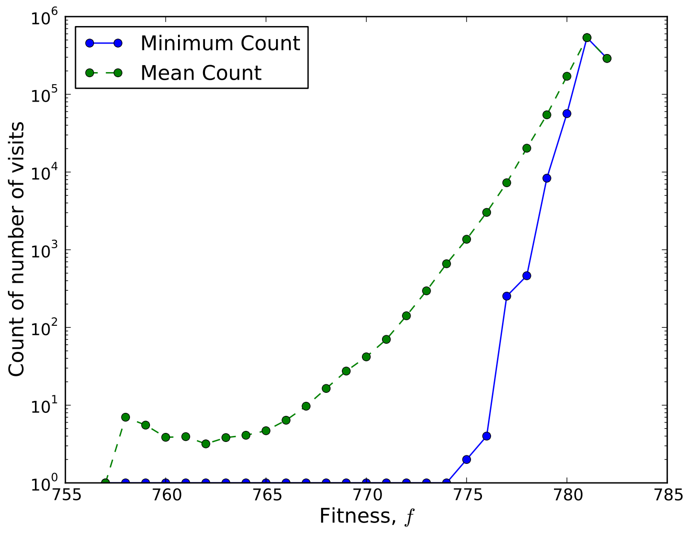


FIGURE 2.9: Number of times local optima are hit in an experiment with 10^7 hill-climbs for a particular randomly drawn instance with $n = 100$ and $\alpha = 8$. This particular instance has a global optimum with an unusually small basin of attraction—hence the kink at high fitness. Also note, at fitness 774 we have visited each local optimum on average around 660 times, but there are 5 local optima at this fitness that have only been visited once.

out the existence of elves empirically, we strongly doubt their existence as we have not found any example of elves in a very large number of trials. Furthermore, as we will see later on, we have strong theoretical reasons to believe that high fitness optima will have relatively large basins of attraction (as is also found empirically), so it is unlikely that there can exist an elf. As a consequence, we believe that we have found all the global optima for the small problem instances we report. Of course, as the problem size increases the number of times we find the fittest observed optima decreases until we can no longer have confidence that a global optimum has not been missed.

Although we can be confident for small instances of reaching all fit local optima, we are almost surely missing local optima with fitness $f < 775$. There exists a number of imputation techniques for estimating the true number of local optima at a given fitness level (see, for example, Garnier and Kallel (2000); Reeves and Eremeev (2004)). A couple of papers have used these methods for estimating the number of local optima for MAX-SAT, but these have concentrated on instances close to the phase transition (Reeves and Aupetit-Bélaidouni, 2004; Albrecht et al., 2010). We have tried applying a simple probabilistic model to our data. We assume that the probability of reaching a local optimum, i, is a random variable, Z_i, where all probabilities of the same fitness, f, come from the same gamma distribution

$$Z_i \sim \gamma(z|a_f, b_f) = \frac{b_f^a z^{a_f - 1} \varepsilon{-b_f z}}{\Gamma(a_f)}.$$

The probability that local optimum, i, is visited n_i times in N trials is assumed to be Poisson distributed

$$\mathbb{P}(n_i|Z_i) = \frac{(NZ_i)^{n_i}\,\varepsilon{-NZ_i}}{n_i!}.$$

The assumption of a Poisson distribution will be reasonably accurate given that each run is independent, and the probability of finishing in any given optimum is small. The choice of a gamma distribution is somewhat arbitrary, although, it is a common choice for modelling distributions of positive random variables, which frequently fits empirical data quite well.

Given a list of visiting frequencies for local maxima with fitness f, $(n_i|f_i = f)$ the likelihood of the data is given by

$$\mathbb{P}((n_i|f_i = f)|a_f, b_f) = \prod_{i\in\{i|f_i=f\}} \int_0^\infty \mathbb{P}(n_i|z_i)\,\gamma(z_i|a_f, b_f)\,\mathrm{d}z_i.$$

We could choose a_f and b_f to maximise this likelihood; however, this fails to take into account those optima that have not been visited. To correct for this, we include n_f^u unobserved local maxima at fitness f in our likelihood estimation, where n_f^u is equal to the expected number of unobserved local optima given a_f and b_f. Details of this calculation are given in appendix B. This inference technique appears to be quite accurate provided the expected number of times the local maxima at fitness f is visited is greater than 1 (this observation is based on running this inference procedure while holding out some of the data—results not shown). When the basins of attraction become smaller than this, the maximum-likelihood estimation breaks down. This is not too surprising as the only data we have to determine the shape of the gamma distribution come from its tail.

Figure 2.10 shows the mean probability of visiting a local optimum of fitness f. The dashed lines show the estimation based on observed local optima (this is just the mean count of the number of visits as shown in Figure 2.9 divided by N). The solid curve shows the estimated values using the maximum-likelihood estimator of a gamma distribution described above. This breaks down when the estimated probability goes below N^{-1} (where in our case $N = 10^7$). We have not shown the maximum-likelihood estimation below this point ($f < 768$) as the algorithm is numerically unstable. Obtaining a reliable estimation for the number of local optima at low fitness is extremely difficult and we have made no further attempt to do so in this research.

In Figure 2.11, we plot the logarithm of the number of observed local maxima divided by n, versus the fitness divided by αn, for three particular instances of size 50, 75 and 100. We performed 10^7 hill-climbs to find the maxima. For $n = 75$ and $n = 100$ we significantly underestimate the true number of local optima at lower fitnesses. The plot shows that these curves are roughly similar for different n. It seems plausible that these curves could converge to a universal curve for sufficiently large n. This behaviour is consistent with the hypothesis that the number of local maxima grows exponentially with the system size.

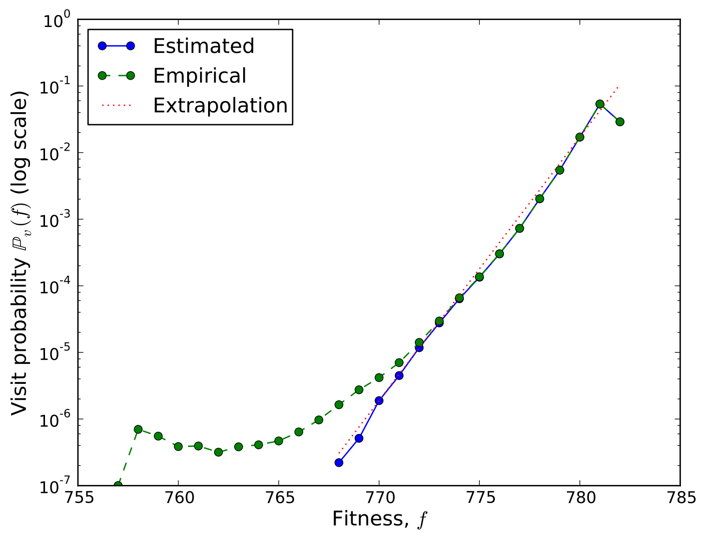


FIGURE 2.10: Mean probability of visiting a local optimum of fitness f using the empirical data (assuming that all local optima have been found) and using the maximum-likelihood estimation described in the text. The extrapolation is just a straight line fit, $\mathbb{P}_v(f) \propto 2.49^f$.

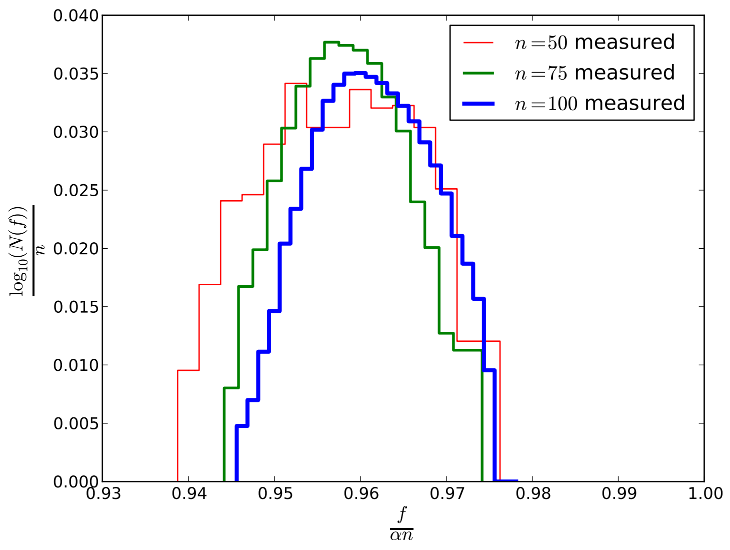


FIGURE 2.11: Plot of the logarithm of the number of local maxima divided by n versus the fitness divided by αn.

The number of local optima also grows with α, although apparently not exponentially. This is shown in Figure 2.12 where we have counted the mean number of local optima found in 10^6 hill-climbs plotted against α. With this number of hill-climbs, some fraction of the local optima will not be found, nevertheless, it is clear that the number of local optima grows, apparently linearly, with α.

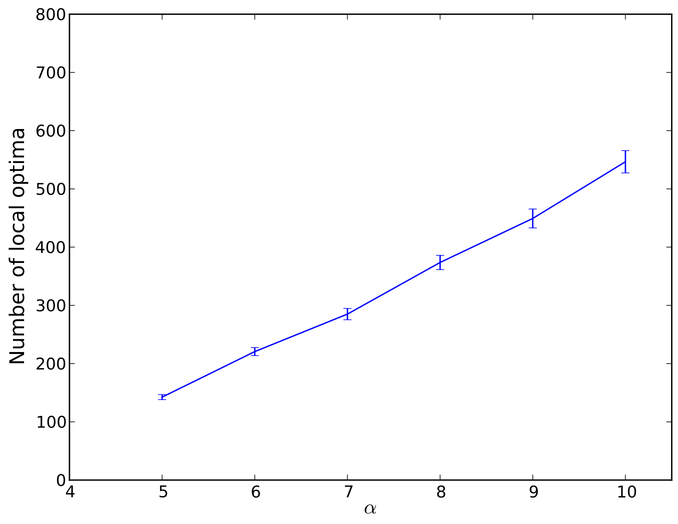


FIGURE 2.12: Mean number of local optima found in 10^6 hill-climbs averaged over 100 instances versus the clause-to-variable ratio, α.

As is evident from Figure 2.10, the expected probability of visiting a local optimum grows as its fitness increases. The ratio of the mean probabilities of visiting a local optimum of fitness $f+1$ to visiting a local optimum of fitness f, is equal to around 2.1 for $n = 50$, 2.47 for $n = 75$, and 2.49 for $n = 100$. A consequence is that there is clearly a strong bias towards high fitness local maxima. Figure 2.13 shows the proportion of local optima at each fitness value, and the probability of finding a local optimum using GSAT. Note that Figure 2.13 shows the empirically measured proportion of local optima at each fitness level and thus underestimates the true proportion of local optima at low fitness.

2.3.5 Reaching the Global Optima

From the perspective of finding fit solutions, it is clearly a desirable property of MAX-SAT that the fitter local optima have, in expectation, larger basins of attraction. However, the gap between the expected fitness found and the fitness of the globally optimal solutions grows with the problem size, n. This is illustrated in Figure 2.14, where we show the maximum fitness and the mean fitness of the local maxima found by GSAT averaged over an ensemble of randomly drawn instances—note that the fitnesses have been scaled by $1/(\alpha n)$ so the gap between the expected fitness found by GSAT and the maximum fitness appears constant for large n. For a single instance, the fitness of the maxima found by GSAT will fluctuate on every run. Figure 2.15

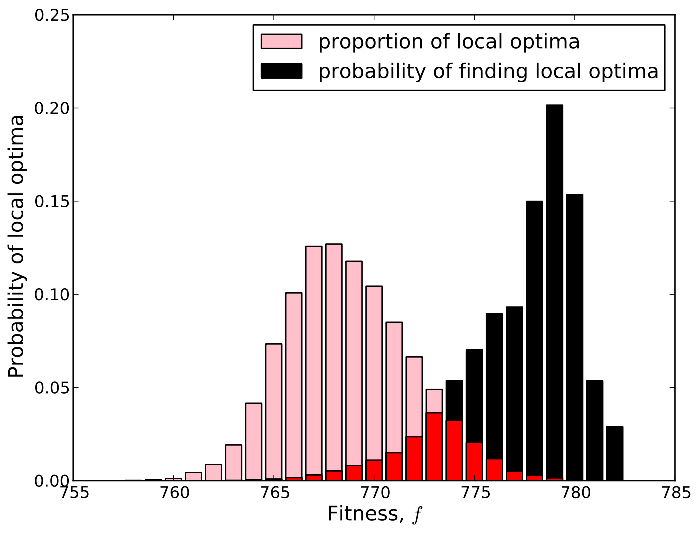


FIGURE 2.13: Histograms showing the proportion of local optima at each fitness value and the probability of finding a local optimum of each fitness, for one instance.

shows the variance in the fitnesses found by different runs of GSAT divided by αn plotted against the reciprocal of the problem size (we used exhaustive search to ensure that we reached a local maximum on each run). The plot is consistent with the variance in fitness growing linearly with the system size n. That is the size of the fluctuations (given by the standard deviation) scales as $\sqrt{n}$. In Figure 2.14, we have also plotted one standard deviation around the average, estimated by extrapolating the empirically measured variance to large n. We see that instances become hard as n becomes large since the gap between the expected fitness of a local optimum found by GSAT and the globally optimal fitness grows linearly with n, while the standard deviation grows only as $\sqrt{n}$. Thus, the chance of finding a global optimum decreases as n increases.

In Figure 2.16, the logarithm of the probability of finding a globally optimal solution is plotted against n. The straight-line fit is consistent with the premise that the probability of finding a global maximum decreases exponentially with the size of the system. Using the straight-line fit in Figure 2.16 to extrapolate to large n, we find the probability of GSAT finding a global optimum for an instance of size 10 000 would be around 10^{-76}. Although the value obtained from such an extrapolation is likely to be inaccurate, nevertheless, it provides a strong indication that the strategy of using multiple runs of GSAT takes exponential time (in the instance size).

2.3.6 Number of Global Optima

Figure 2.17 shows a histogram of the number of global optima for 10 000 random MAX-3-SAT instances with $n = 100$ and $\alpha = 8$. We observe that there is quite a wide spread in the number of the global optima. The expected number of global optima raises from 2.7 for $n = 40$ to around 4.0 for $n = 200$.

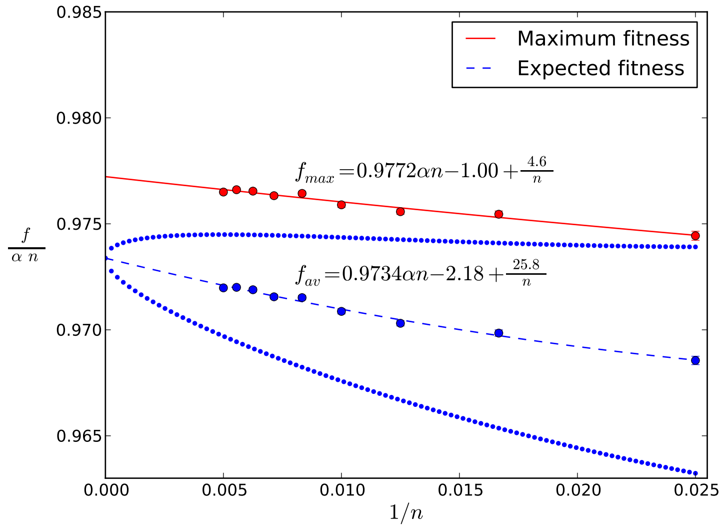


FIGURE 2.14: Plot of the expected maximum fitness and the average fitness found by GSAT versus $1/n$. This is calculated by performing 10^5 hill-climbs on 400 randomly generated instances of the problem. We have computed the standard deviation in the fitness found by GSAT and show the mean plus and minus 1 standard deviation extrapolated for all n by the dotted lines.

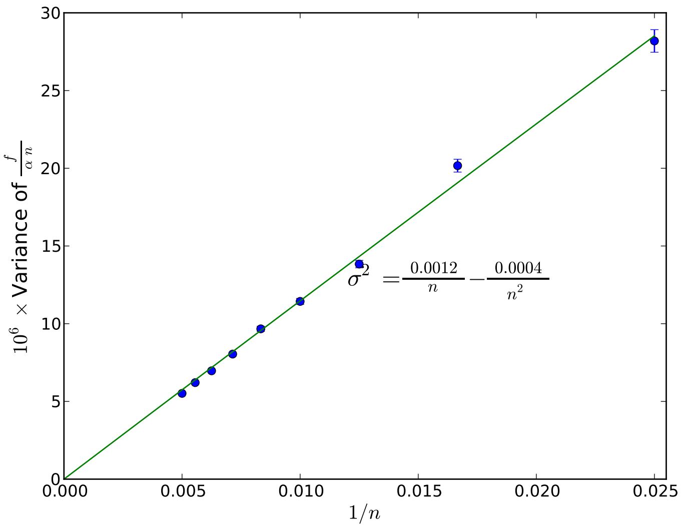


FIGURE 2.15: Plot of the variance of $f/(\alpha n)$ found by GSAT averaged over 400 instances against $1/n$.

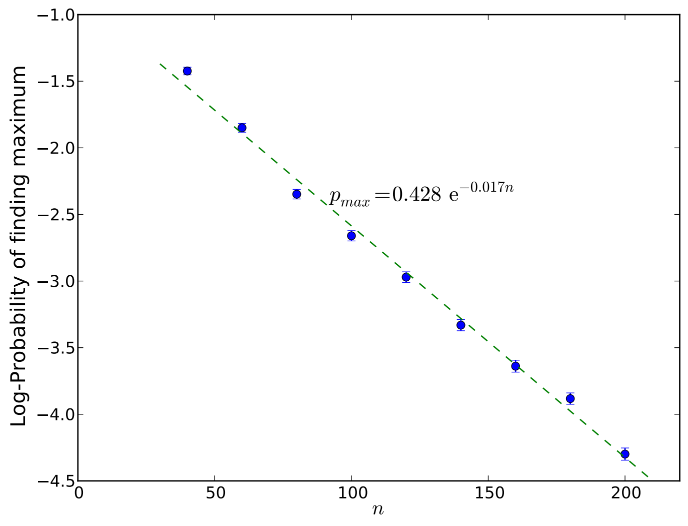


FIGURE 2.16: Natural log-probability of finding the maximum fitness optima versus n averaged over 400 randomly generated instances.

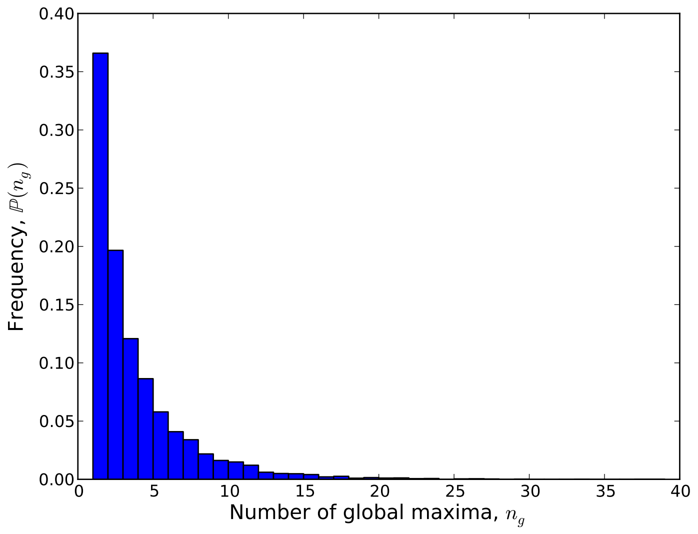


FIGURE 2.17: Histogram of the number of global optima for 10 000 random MAX-3-SAT instances with $n = 100$ and $\alpha = 8$.

Note that we define each optimum to be a connected set of configurations at the same fitness with no fitter neighbours (see Figure 1.1). Figure 2.18 shows a histogram of the number of configurations in each global optimum measured in 10 000 randomly generated problems. The tail of the distribution has been truncated in Figure 2.18; there are 468 global optima with more than 200 configurations. The largest global optimum in these 10 000 instances has 2136 configurations.

In Figure 2.19 the same data used in Figure 2.18 is plotted on log-log axes. We see, for large n_f, that $\mathbb{P}(n_f)$ falls off close to $1/n_f$. This is an extremely fat-tailed distribution for which there is

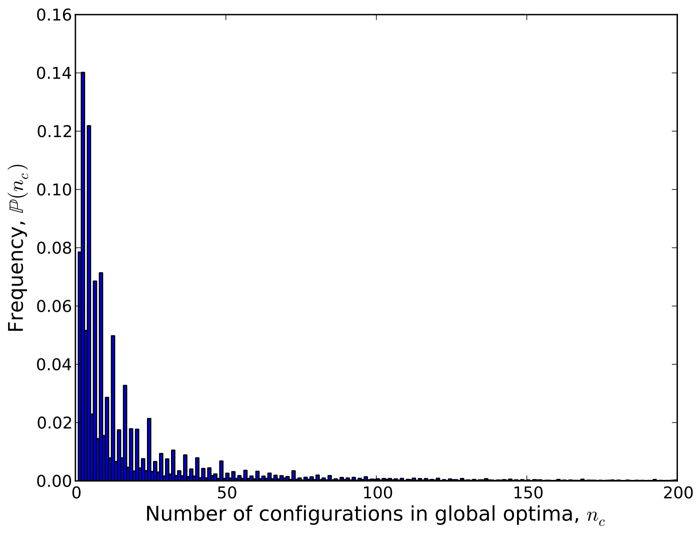


FIGURE 2.18: Histogram of the number of configurations in each global optimum for 10 000 randomly generated problems with $n = 100$ and $\alpha = 8$. The histogram has been truncated with 468 global optima having more than 200 configurations and where the maximum is 2136.

no mean. It is interesting to observe that although these instances are in many ways statistically similar, at least, in regard to the size of the global maxima the instances can have dramatically different properties.

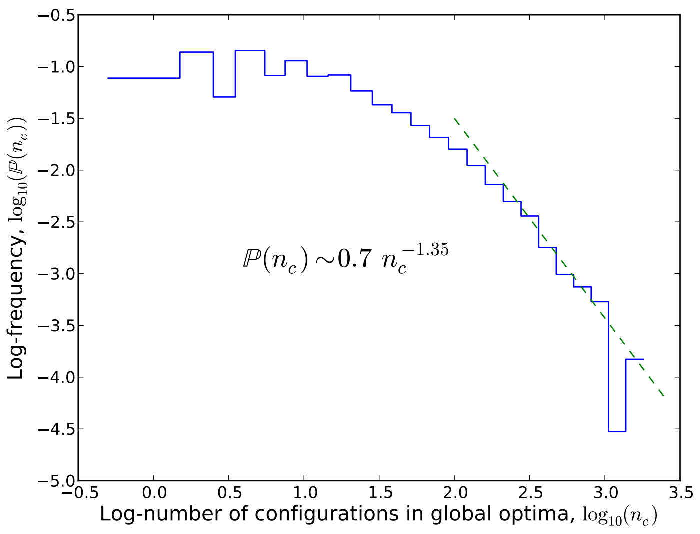


FIGURE 2.19: Distribution of the size of global optima plot on a log-log axis. To plot this, the data in Figure 2.18 has been put into bins of increasing size. The dashed line is a straight line fit corresponding to a distribution with a tail falling off as $0.7 n_c^{-1.35}$.

The distance between global optima also varies from instance to instance. Figure 2.20 shows schematically the sizes of the global optima and the mean and minimum Hamming distances between the optima for one particular instance with three global optima. Figure 2.21 shows a histogram of Hamming distances between all configurations that make up the global optima for this particular instance.

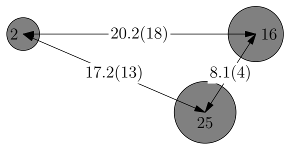


FIGURE 2.20: Pictorial representation of the distance between the global optima in a particular instance of a problem with $n = 100$ and $\alpha = 8$. The number of configurations in the three global optima are 2, 16 and 25 while the average Hamming distance between global optima are 20.2, 17.2 and 8.1 with the minimum Hamming distance being 18, 13 and 4.

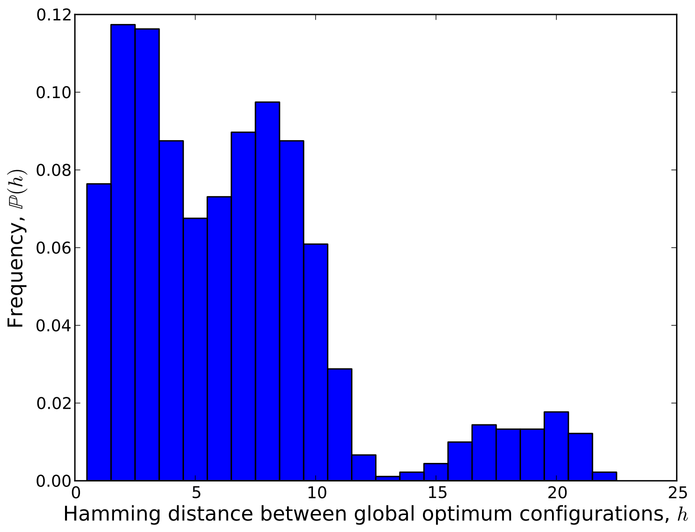


FIGURE 2.21: Histogram of Hamming distances between configurations at the global optimal fitness for the same instance as shown in Figure 2.20.

In Figure 2.22 we show a histogram of the Hamming distances between configurations at the global optimal fitness, averaged over 1000 instances, with $n = 100$ and $\alpha = 8$. The peak at low values is produced predominantly by the configurations in the same global optimum, although there are also global optima with a minimum Hamming separation of 2. Notice that there is a wide spread of Hamming distance with some global optima having a Hamming distance greater than $n/2$. This may appear counter-intuitive as it shows that there are instances where there exist entirely unrelated ways of optimally solving these problems. However, there are $2^\beta - 1 = 7$ different ways in which each clause can be satisfied. Since the instances are constructed from random clauses, each variable occurs in roughly the same number of clauses, with the variables being negated roughly half the time. In a fit configuration, the truth values of the variables are chosen with respect to each other so that a large number of clauses are satisfied, but this careful balance can be achieved in many different ways. Thus, it is not too difficult to understand why there can be global optima which are a considerable Hamming distance apart. For more structured MAX-SAT problems, the balance of variables in each clause may be decidedly different so that good solutions may be more correlated. An early observation about SAT and MAX-SAT was that most instances were found to be easy to solve. Later it was found that random instances seem particularly hard (Mitchell et al., 1992). A consequence of the structure of random instances of MAX-SAT is that a slight change to the set of clauses, although only resulting in a small change in the fitness for each configuration, can totally change the ranking of the local optima, and so may swap the position of the global optima to a quite different part of the search space. This sensitivity makes it very difficult to construct a heuristic for finding a global optimum.

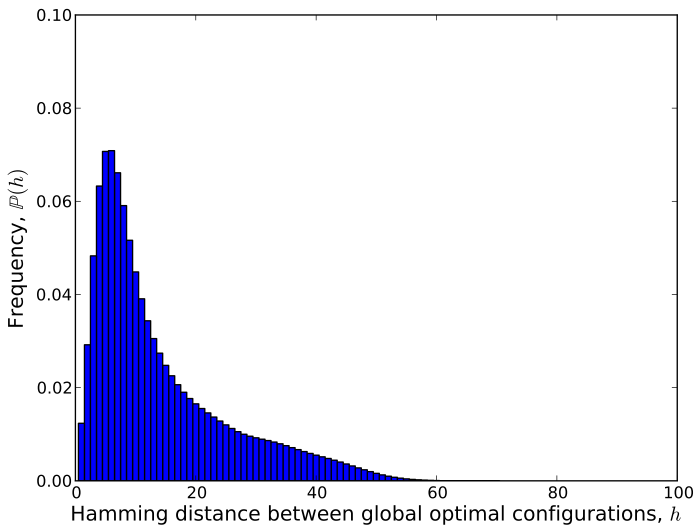


FIGURE 2.22: Histogram of Hamming distances between configurations at the global optimal fitness for 1000 instances of MAX-3-SAT with $n = 100$ and $\alpha = 8$.

2.3.7 Distance between Optima

We have seen that, for large instances, GSAT will, with overwhelming probability, finish in a local optimum with a fitness considerably lower than the maximum fitness. This might not prevent an algorithm from finding a globally optimal solution, provided the global optimum was close to most good local optima. If this was the case, it might be possible to develop an algorithm which quickly moved from one local optimum to a fitter one. However, we show here that this hope is fruitless. Figure 2.23 shows the mean Hamming distance from a configuration in a local optimum to the *nearest* global optimum. These results are averaged over all local optima found in 100 instances.

We can collapse the curves in Figure 2.23 onto a universal curve by rescaling the axes and fitting a single correction to scaling parameter. This is shown in Figure 2.24 and demonstrates that the distance from a local optimum to a global optimum scales linearly with the problem size. The Hamming distance between an optimum of fitness $f_{max} - 1$ to f_{max} clearly grows with increased n, although it is difficult to be sure how this Hamming gap grows. The answer to this question depends on extrapolating the curve in Figure 2.24 to the y-axis. A plausible

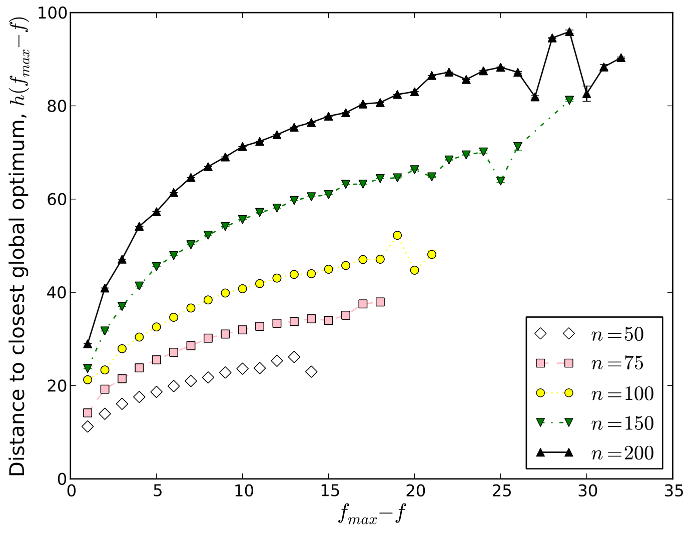


FIGURE 2.23: Measure of the mean Hamming distance to the closest global optimum from a local optimum of fitness f.

extrapolation would be that this gap was around $0.1n$, although it is not inconceivable that it extrapolates to zero indicating that the Hamming distance would be $o(n)$ and $\omega(1)$. As the number of configurations in a Hamming-ball of radius h grows as $\binom{n}{h} = \varepsilon\Theta(h)$ (for $h > n/2$), the expected time to move from a sub-optimal local maximum to an optimal local maximum by searching the neighbourhood would (extrapolating from this data) appear to increase super-polynomially.

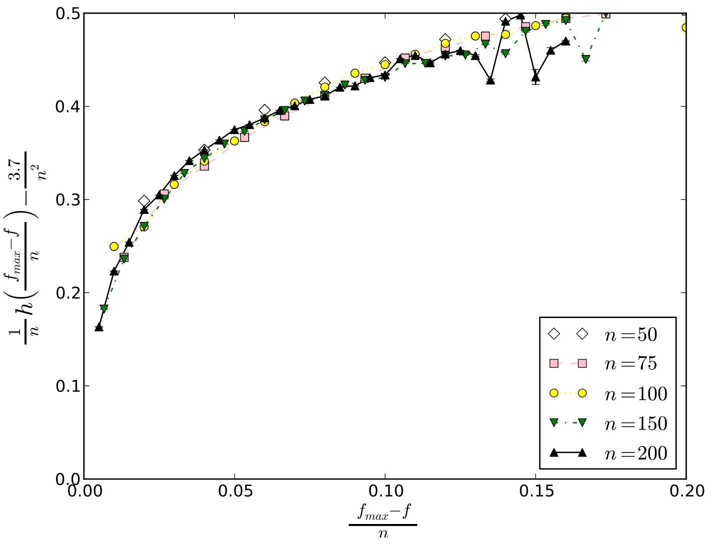


FIGURE 2.24: Rescaled version of Figure 2.23 showing that the curves collapse on to each other. To get a decent collapse we included a "correction to scaling" term ($-3.7/n^2$), where the parameter was chosen to fit the data.

2.4 Landscape Correlations

The evidence presented thus far provides compelling reasons for believing that the landscape is difficult for a hill-climber to navigate. However, as we saw in the auto-correlation there exist significant long-range correlations in the landscape. This is a consequence of the structure of the objective (fitness) function. On average, each variable occurs in just $\beta \times \alpha$ clauses, with an equal probability of a clause depending on the variable or on its negation. The change in fitness due to flipping a variable will therefore be equal to the sum of the clauses that are satisfied only by that variable minus the sum of clauses that are unsatisfied, but are made satisfied by flipping the variable. Typically, this change in fitness is quite small. As a consequence, the configurations around a fit configuration also tend to be fit.

2.4.1 Expected Fitness in Hamming Sphere

We can analytically compute the mean fitness in a Hamming sphere around any configuration from knowledge of the number of satisfied literals in each clause. Let us denote the variables by $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and the clauses by $g_i(\boldsymbol{x})$ where $i = 1, 2, \ldots, m = \alpha n$. We partition the clauses (for a given configuration $\boldsymbol{x}$) into equivalence classes, $\mathscr{S}_l$, depending on the number of satisfied literals in the clause. Thus, we write $g_i(\boldsymbol{x}) \in \mathscr{S}_l$ if clause i has l satisfied literals. Clearly l can take values 0 to β (the number of literals in each clause). We denote the indicator function using square brackets $[\![\textit{predicate}]\!]$, which is equal to 1 if the *predicate* is true and 0 otherwise. We denote the size of the equivalence classes (i.e., the number of clauses with l satisfied literals) by

$$s_l(\boldsymbol{x}) = \sum_{i=1}^{m} [\![g_i(\boldsymbol{x}) \in \mathscr{S}_l]\!] .$$

Since the fitness, $f(\boldsymbol{x})$, of a configuration is equal to the number of satisfied clauses we have

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{i=1}^{m} \sum_{l=1}^{\beta} [\![g_i(\boldsymbol{x}) \in \mathscr{S}_l]\!] = m - \sum_{i=1}^{m} [\![g_i(\boldsymbol{x}) \in \mathscr{S}_0]\!] \\ &= m - s_0(\boldsymbol{x}). \end{aligned}$$

To compute the expected fitness in a Hamming sphere of radius h we average over all configurations $\boldsymbol{x}'$ with Hamming distance $d(\boldsymbol{x}', \boldsymbol{x}) = h$ from the configuration of interest, $\boldsymbol{x}$. We denote the set of configurations in this Hamming sphere by

$$\mathscr{X}_h(\boldsymbol{x}) = \left\{ \boldsymbol{x}' \in \{T, F\}^n | d(\boldsymbol{x}', \boldsymbol{x}) = h \right\} .$$

We note that $|\mathscr{X}_h(\boldsymbol{x})| = \binom{n}{h}$. The expected fitness in the Hamming sphere is

$$f_h(\boldsymbol{x}) = \mathbb{E}\left(f(\boldsymbol{x}') | d(\boldsymbol{x}', \boldsymbol{x}) = h \right) = m - c_h(\boldsymbol{x}),$$

where $c_h(\boldsymbol{x})$ is the expected number of unsatisfied clauses (or cost) at a Hamming distance h from configuration $\boldsymbol{x}$,

$$\begin{aligned} c_h(\boldsymbol{x}) &= \mathbb{E}\left(s_0(\boldsymbol{x}')|d(\boldsymbol{x}',\boldsymbol{x}) = h\right) \\ &= \frac{1}{\binom{n}{h}} \sum_{\boldsymbol{x}' \in \mathscr{X}_h(\boldsymbol{x})} \sum_{i=1}^{m} [\![g_i(\boldsymbol{x}') \in \mathscr{S}_0]\!] \,. \end{aligned}$$

The number of unsatisfied literals in any clause must be in the set $\{0, 1 \ldots, \beta\}$ so that

$$\sum_{l=0}^{\beta} [\![g_i(\boldsymbol{x}) \in \mathscr{S}_l]\!] = 1.$$

Putting this into the equation for $c_h(\boldsymbol{x})$ we obtain

$$c_h(\boldsymbol{x}) = \frac{1}{\binom{n}{h}} \sum_{l=0}^{\beta} \sum_{i=1}^{m} \sum_{\boldsymbol{x}' \in \mathscr{X}_h(\boldsymbol{x})} [\![g_i(\boldsymbol{x}') \in \mathscr{S}_0]\!] \, [\![g_i(\boldsymbol{x}) \in \mathscr{S}_l]\!] \,,$$

where we have reordered the summations (which we are clearly allowed to do as they are all over finite ranges). Now we observe that

$$\sum_{\boldsymbol{x}' \in \mathscr{X}_h(\boldsymbol{x})} [\![g_i(\boldsymbol{x}') \in \mathscr{S}_0]\!] \, [\![g_i(\boldsymbol{x}) \in \mathscr{S}_l]\!] = \binom{n-\beta}{h-l} [\![g_i(\boldsymbol{x}) \in \mathscr{S}_l]\!] \,,$$

since, for the indicator functions to be true, we have to flip the l satisfied variables in clause i and leave the unsatisfied variables unchanged. This means we have to flip $h-l$ variables that are not in clause i (there are $n-\beta$ such variables). Thus, there are $n-\beta$ choose $h-l$ ways to flip these variables. Substituting this result into our expression for $c_h(\boldsymbol{x})$ we find

$$\begin{aligned} f_h(\boldsymbol{x}) &= m - \frac{1}{\binom{n}{h}} \sum_{l=0}^{\beta} \sum_{i=1}^{m} \binom{n-\beta}{h-l} [\![g_i(\boldsymbol{x}) \in \mathscr{S}_l]\!] \\ &= m - \frac{1}{\binom{n}{h}} \sum_{l=0}^{\beta} \binom{n-\beta}{h-l} s_l(\boldsymbol{x}). \end{aligned}$$

This is easy to compute given $s_l(\boldsymbol{x})$ for $l = 0, 1, \ldots, \beta$. Obtaining the expected variance in the same Hamming sphere is much more complicated than the mean fitness as it depends on the similarity between clauses. We can, however, obtain a simple approximation for the variance—details are given in appendix C.

In passing, we note that Grover (1992) defined a difference operator for a neighbourhood $\mathscr{N}(\boldsymbol{x})$ as

$$\nabla^2 f(\boldsymbol{x}) = \frac{1}{|\mathscr{N}(\boldsymbol{x})|} \sum_{\boldsymbol{x}' \in \mathscr{N}(\boldsymbol{x})} \Big(f(\boldsymbol{x}') - f(\boldsymbol{x}) \Big) \,,$$

and defined a landscape to be *elementary* if it satisfies the wave-like equation

$$\nabla^2 f(\boldsymbol{x}) = \lambda f(\boldsymbol{x}),$$

for some λ. For a Hamming neighbourhood $\nabla^2 f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f(\boldsymbol{x})$, where for MAX-SAT, $f_1(\boldsymbol{x})$ is equal to

$$\begin{aligned} f_1(\boldsymbol{x}) &= m - \frac{1}{n}\Big((n-\beta)s_0(\boldsymbol{x}) + s_1(\boldsymbol{x})\Big) \\ &= \frac{\beta m}{n} + \frac{(n-\beta)}{n} f(\boldsymbol{x}) - \frac{s_1(\boldsymbol{x})}{n}, \end{aligned}$$

so that

$$\nabla^2 f(\boldsymbol{x}) = \frac{\beta}{n}(m - f(\boldsymbol{x})) - \frac{s_1(\boldsymbol{x})}{n}.$$

The dependence of this on $s_1(\boldsymbol{x})$, which differs from configuration to configuration, prevents MAX-SAT from being an elementary landscape. However, Max-β-Sat has been shown to be a superposition of β-elementary landscapes, so it has some intriguing algebraic properties (Sutton et al., 2009).

Figure 2.25 shows a bar chart of the number of clauses with l satisfied literals (l = 0, 1, 2 and 3), for a configuration in the global optimum with the largest basin of attraction for a particular instance. Similar qualitative features are observed for all fit configurations.

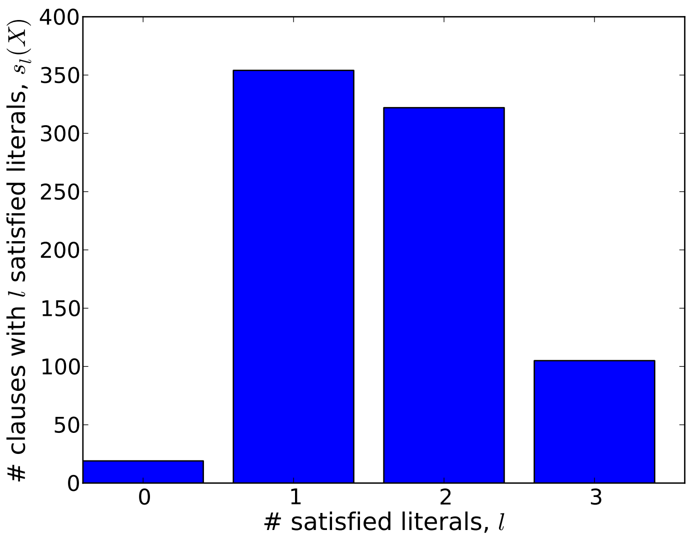


FIGURE 2.25: Number of clauses with l satisfied literals is shown for one of the optimal configurations of a MAX-3-SAT problem with $n = 100$ and $\alpha = 8$.

In Figure 2.26, we show the expected fitness of a configuration in a Hamming sphere of radius h from the configuration, $\boldsymbol{x}$, with $s_l(\boldsymbol{x})$ given in Figure 2.25. The behaviour of the curve at large Hamming distances is predominantly determined by the number of configurations where all the literals are satisfied (i.e., $s_3(\boldsymbol{x})$)—this number is not terribly consistent between optima of the same fitness.

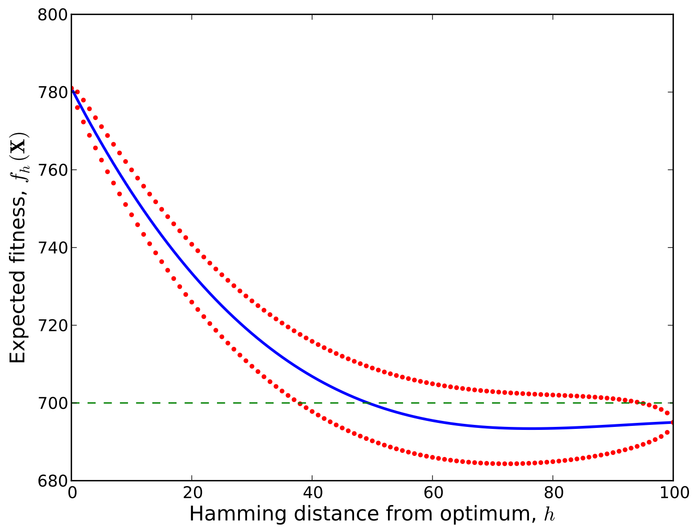


FIGURE 2.26: Expected fitness of configurations in a Hamming sphere of radius h around the same configuration shown in Figure 2.25. The dotted curves show one standard deviation around the mean. Note that the average fitness (shown by the horizontal dashed line) is at $(1-2^{-\beta})\alpha n = 700$.

The qualitative shape of the curve is similar for all n, although the standard deviation in the fitness is of order $\sqrt{n}$. In Figure 2.27 we show the expected fitness for an instance with $n = 10000$ and $\alpha = 8$ around a good solution found using landscape guided hopping (Qasem and Prügel-Bennett, 2010). The solution is extremely unlikely to be a global optimum and may not be a local optimum.

If we scale the fitness by the system size we observe that the line for the average fitness is very similar to that of Figure 2.26, although, the relative variance is clearly much smaller. Figure 2.28 shows the expected fitness for each Hamming sphere for the most frequently visited optima at each fitness where an optimum was found. At each fitness level, we chose the most frequently visited optimum. We note the same qualitative behaviour in all the curves, although there is some slight variation. A few of the curves cross each other, and differ markedly for large Hamming distances.

These curves show the existence of large-scale correlations in the fitness landscape. That is, the presence of a local optimum changes the expected fitness of configurations away from the mean fitness at all Hamming distances. Despite this large-scale structure, local-search algorithms still fail to reliably find a global optimum. The reason for this is that the landscape is sufficiently rugged that it is not possible to exploit the long-range correlation using local fitness information alone. Figure 2.29 shows a density plot of the local optima as a function of their fitness and their Hamming distance from the most frequently visited global optimum. There is a small

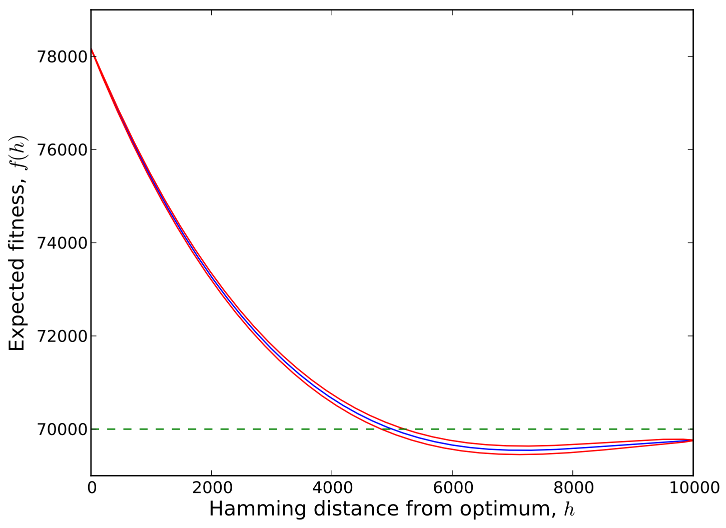


FIGURE 2.27: Expected fitness of configurations in a Hamming sphere of radius h around a high-fitness configuration in a problem with $n = 10\,000$ and $\alpha = 8$. The number of clauses with l satisfied literals are $s_0(\boldsymbol{x}) = 1\,830$, $s_1(\boldsymbol{x}) = 36\,842$, $s_2(\boldsymbol{x}) = 31\,086$ and $s_3(\boldsymbol{x}) = 10\,242$.

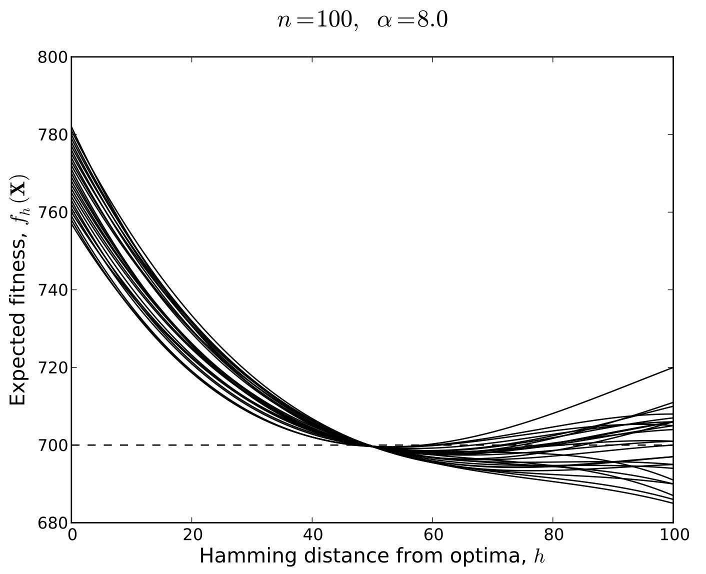


FIGURE 2.28: Expected fitness in a Hamming sphere for the most frequently visited optima at each fitness where an optimum was found. This is for the same instance as that shown in Figure 2.26.

correlation between the quality of the local optima and the closeness to this global optimum, however, the vast majority of local optima are around $n/2$.

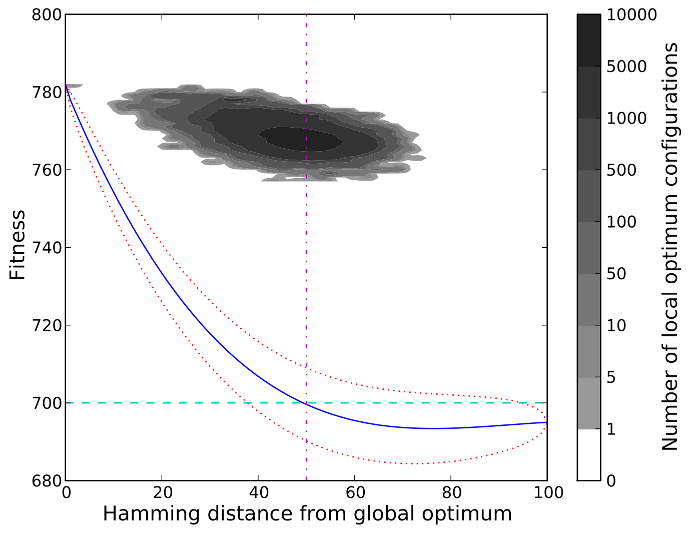


FIGURE 2.29: Density plot of local optimum configurations as a function of their fitness and Hamming distance from the most frequently visited global optimum. We also plot the mean fitness in a Hamming sphere from the same global optimum. The shading of the point shows the number of configurations at that fitness and Hamming distance. Note the exponential scale on the density. We only show those local optimum configurations found using 10^7 hill-climbs, so we are likely to have missed many low-fitness optima.

A problem instance would satisfy the big-valley hypothesis if the closer a local optimum is to the global optimum the fitter the optimum (Boese et al., 1994a; Boese, 1995). In MAX-3-SAT we often have multiple global optima, which, as shown in Figure 2.22, can be a considerable distance apart. Furthermore, although there is a slight tendency for fit local optima to be close to a global optimum, there are plenty of fit local optima at a considerable distance from a global optimum, and unfit local optima close to a global optimum. Thus, these instances do not have a classic big-valley structure.

2.4.2 Basins of Attraction

We can empirically measure the "basin of attraction" of a local optimum by repeatedly flipping a fixed number of variables and running GSAT until it reaches a local optimum. We then see if this is the same local optimum from which we started. We can thus measure the return probability starting from a configuration in a given Hamming sphere. Note that since GSAT is stochastic the basin of attraction is a probabilistic concept even at the level of individual configurations. Figure 2.30 shows the return probability for the most frequently visited global optimum. We observe that we have above a 50% chance of reaching the global maximum if we start within a Hamming distance of around 30. Figure 2.31 shows the return probability for one of the local optima with the smallest found fitness for the same instance as that shown in Figure 2.30. This local optimum was found only once in 10^7 hill-climbs.

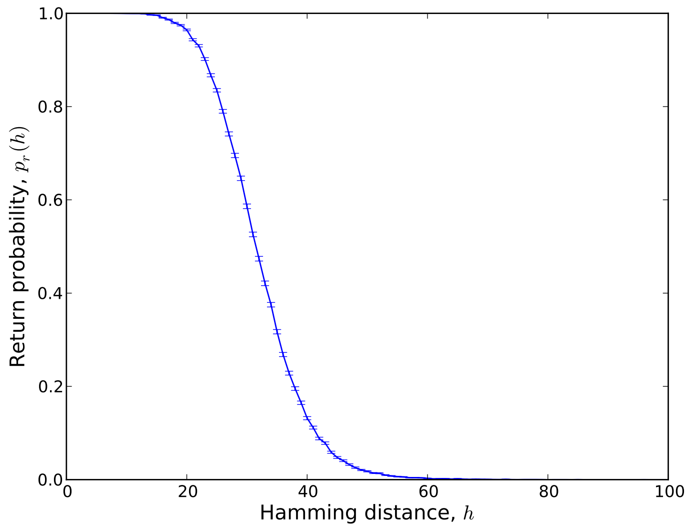


FIGURE 2.30: Return probability starting from a Hamming sphere of radius h versus h. This is for the most probable global optimum solution of an instance with $n = 100$ and $\alpha = 8$. It has a probability of being visited of 0.029. The empirical probability is computed by running 10 000 hill-climbs starting at randomly chosen configurations in each Hamming sphere.

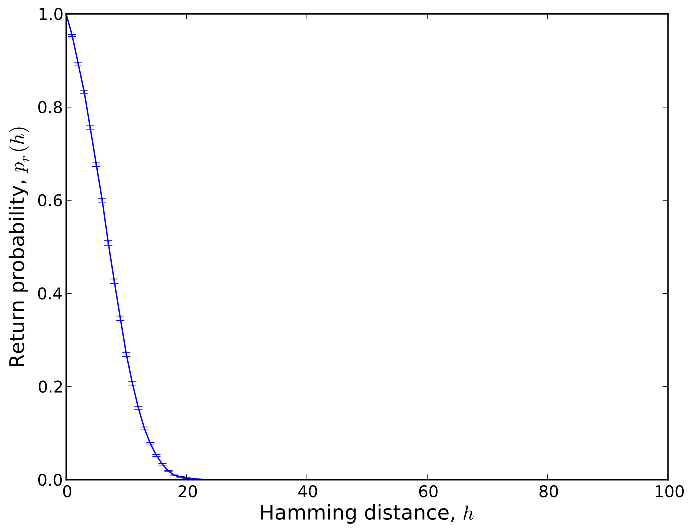


FIGURE 2.31: Return probability starting from a Hamming sphere of radius h versus h. This is for a local optimum with the smallest fitness that we found. The local optimum was found once in 10^{-7} hill-climbs.

In principal, given the return probability, $p_r(h)$, it is straightforward to compute the probability of finding the local optimum starting from a random initial position. This is given by

$$\mathbb{P}\,(\text{finding local optimum}) = \frac{1}{2^n}\sum_{h=0}^{n}\binom{n}{h}\,p_r(h),$$

since the probability of starting in a Hamming sphere of radius h is $2^{-n}\binom{n}{h}$. Unfortunately, for return probabilities that fall off rapidly, this sum is dominated by the tail of the distribution which is effectively truncated when we measure $p_r(h)$ empirically. Using the empirically measured values of $p_r(h)$ thus severely underestimates the probability of reaching an optimum for optima with small basins of attraction. We can obtain a better estimate of the probability for finding a local maximum by approximating the tail of $p_r(h)$ by the best-fit exponential. Using this estimate we find the probability of finding the local maximum shown in Figure 2.31 is approximately 1.3×10^{-9}. Given that we only sampled 10^7 times it might appear that we were lucky to find a maximum with such a small basin of attraction. However, there are presumably so many such maxima that we are exceedingly likely to find at least one.

2.5 Conclusion

The picture of how MAX-3-SAT becomes difficult is known to be qualitatively similar to a large number of other hard optimisation problems. The analysis presented in this thesis fills in the details. We see that there is an exponential growth in the number of local optima. Although, fitter maxima tend to have larger basins of attraction, this bias is not sufficient to out-weigh the raise in the number of local optima. The fitness gap between a typical local optimum found by GSAT and the global optima increases with n. The fluctuations between runs grow as $\sqrt{n}$ so the chances of reaching a global optimum decrease. The Hamming distance between local and global optima also seem to grow with n, so finding a fit local optimum does not provide significant information about the location of a global optimum.

WALKSAT improves on GSAT by making the search more stochastic. This seems to allow WALKSAT to discover fitter local optima with larger basins of attraction. Finally, we can take advantage of the correlation between fit configurations. This correlation follows from the fact that the average fitness around any configuration will change slowly with the distance from that configuration. We can use this to make large, beneficial hops across the search space by moving to the closest configuration to a set of fit solutions. This averaging is a rather crude way to hop over the search space, and there may well be better ways to exploit knowledge of the average large-scale structure.

Chapter 3

Graph Colouring

The fitness landscape of the Graph-Colouring problem is studied in this chapter. We start with defining the problem, the local search operators and then we move to studying different properties of the fitness landscape.

3.1 Problem Definition

In this section, we describe the graph-colouring problem and the way we generate the problem instances. We finish this section with a discussion about the local-search algorithm we use to find the local optima, and the way we distinguish different local optima from one another.

The graph-colouring problem is a combinatorial optimisation problem which belongs to the class of NP-Hard problems. Given an undirected graph $G(\mathscr{V},\mathscr{E})$, with a vertex (node) set $\mathscr{V}$ and edge set $\mathscr{E}$, and k different colours, the graph-colouring problem is defined as finding a colouring of the vertices to minimise the number of edges whose vertices share the same colour. We denote a configuration of the graph-colouring problem with k colours as a vector $\boldsymbol{x}$ of size $n = |\mathscr{V}|$, with elements $x_i \in \{1, 2, \ldots, k\}$ representing the colour of the i-th node. The cost of a configuration $\boldsymbol{x}$ is defined as the number of *colour conflicts* in the graph, i.e., the number of edges whose vertices have identical colours. That is,

$$c(G,\boldsymbol{x}) = \sum_{(i,j)\in\mathscr{E}} [\![x_i = x_j]\!] ,$$

where $[\![predicate]\!]$ denotes the indicator function that is equal to 1 if the *predicate* is true and 0 otherwise. The chromatic number, $\chi(G)$ of a graph, G, is defined to be the smallest number of colours k such that a configuration exists that has no colour conflicts (i.e., a cost of zero).

The graph-colouring problem is usually posed in three different ways.

1. Given a graph G, find the chromatic number of the graph $\chi(G)$, which belongs to the class of NP-Hard problems.
2. Given a graph G and k different colours, deciding if the graph is colourable with these colours. This problem belongs to the class of NP-Complete problems.
3. Given a graph G and k finding the solution $\boldsymbol{x}$ that has the minimum cost. This problem belongs to the class of NP-Hard problems.

Our interest will focus on this final viewpoint.

In this research we will concentrate on instances drawn from the ensemble of random graphs $\mathscr{G}(n,p)$, consisting of graphs with n vertices where each edge is drawn with a probability p. In particular we focus on the case $p = 0.5$, so that the graphs are dense in the sense that typically the number of edges is of order n^2. For these instances we study the properties both as a function of the number of vertices n and as a function of the number of colours k.

3.2 Colour Symmetry and Distance Measures

An important feature of the graph-colouring is that if we permute all the colours, then the cost is unchanged. As there are $k!$ permutations of the colours, there is a $k!$-fold symmetry in the search space. Graph-colouring can also be viewed as a partitioning problem, where we try to partition the vertices into k partitions so as to minimise the number of edges with vertices in the same partition. In this partition view of the problem we eliminate the $k!$ symmetry of the problem. Although it is more logical to view graph-colouring as a partitioning problem, most algorithms treat the problem as a colouring problem. This reflects the fact that partitions are difficult to treat (e.g. it is non-trivial to determine whether two partitions are identical). In this research, we have tried to accommodate both views of the problem: either as a colouring problem with a $k!$-fold symmetry or as a partition problem.

As a consequence of these two views of the problem we consider two distance measures between configurations. The first is the Hamming distance defined as,

$$D_h(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{n} [\![x_i \neq y_i]\!] \,. \tag{3.1}$$

The second measure is a measure of the ‘partition distance’ defined as,

$$D_p(\boldsymbol{x},\boldsymbol{y}) = \min_{\pi} D_h(\boldsymbol{x}, \pi\boldsymbol{y}), \tag{3.2}$$

where $\pi(\cdot)$ is a permutation operator that permutes the colours. The minimisation is over all possible permutations of the k colours. The partition distance measures the smallest number of reallocations of partition membership to make the partition represented by $\boldsymbol{x}$ into the partition represented by $\boldsymbol{y}$. When the Hamming distance is small, it is often the same as the partition distance. In practice, we can compute the partition distance in $O(k^3)$ by representing the colour matching as a linear assignment problem and using the Hungarian algorithm (Glass and Prügel-Bennett, 2005).

It is useful to understand the distribution of distances between random configurations. Note that this property depends only on the number of vertices, n, and number of colours, k, but is otherwise independent of the problem instances. For two randomly generated configurations the probability that the Hamming distance is equal to h is given by a binomial distribution,

$$\mathbb{P}\left(D_h(\boldsymbol{x},\boldsymbol{y}) = h\right) = \binom{n}{h}\left(1-\frac{1}{k}\right)^h\left(\frac{1}{k}\right)^{n-h},$$

so that the expected Hamming distance between randomly chosen configurations is,

$$E(D_h(\boldsymbol{x},\boldsymbol{y})) = n\left(1-\frac{1}{k}\right).$$

We are not aware of an analytic formula for the probability distribution of partition distances between randomly drawn configurations. In Figure 3.1-b we show the probability distribution of distances between randomly chosen pairs of configurations. The partition distance is measured empirically by sampling.

The average partition distances between the solutions in the search space for different values of n and k are shown in Figure 3.1-b. In this Figure, the horizontal axis is k and the vertical axis is the expected partition distance divided by n. Note that the partition distance can be viewed as the minimum distance between a configuration $\boldsymbol{x}$ and a set of configurations $\pi(\boldsymbol{y})$. The average partition distance is a result of two competing effects. As we increase k the average Hamming distance between random configurations increases as $n(1-1/k)$. As a consequence initially the average partition distance increases. However, as k increases, the size of the permutation symmetry increases as $k!$; thus the likelihood of there being a closer configuration in the set of permuted configurations increases. As a result, the average partition distance begins to fall again. We can obtain very similar qualitative behaviour by considering the expected minimum Hamming distance between a configuration and a set of $k!$ random configurations.

3.3 Chromatic Number

The chromatic number of a graph is the smallest number of colours needed to colour the graph with no colour conflicts (i.e., there exists a zero-cost colouring). Random graphs drawn from the ensemble $\mathscr{G}(n,p)$ can have different chromatic numbers; however, it is found that the chromatic number tends to be highly concentrated. This is illustrated in Figure 3.2-a, where we show the proportion of graphs that are colourable with k, for graphs drawn from $\mathscr{G}(n,p)$ with $n = 10$, 50 and 100. These results were found empirically, by randomly drawing samples and repeatedly

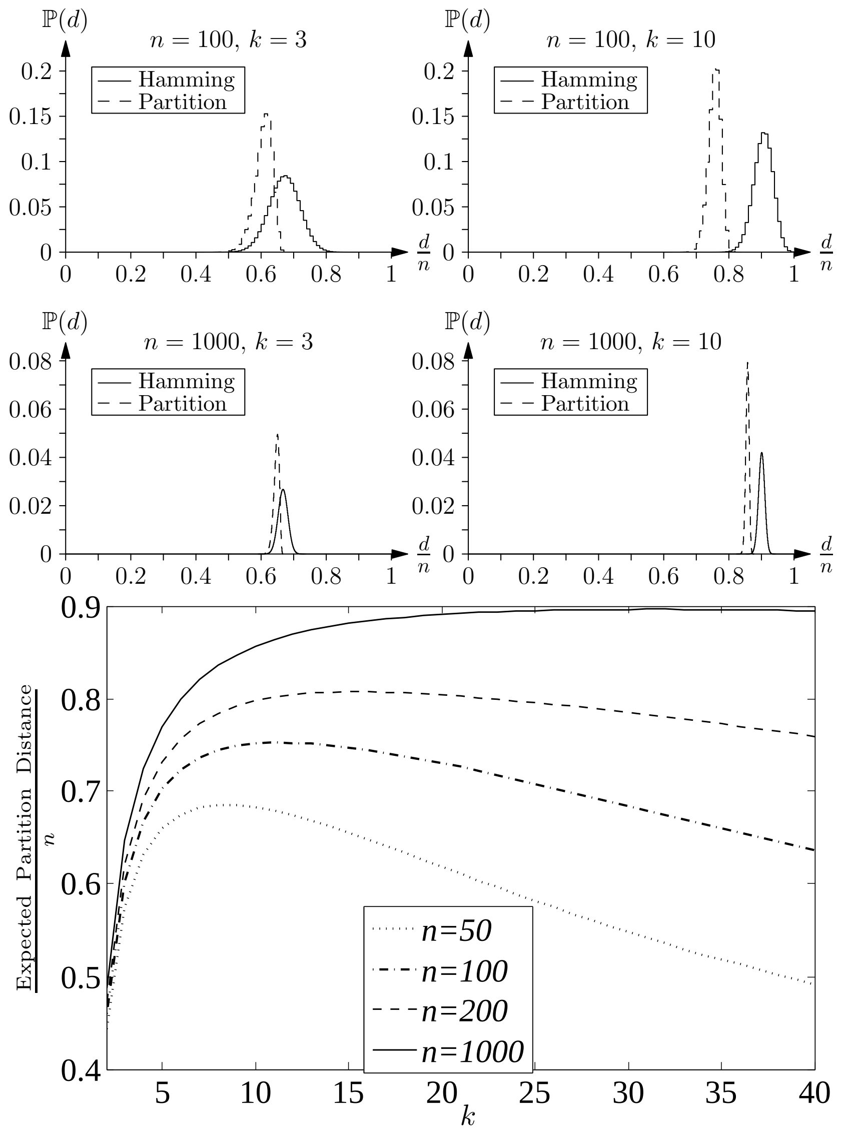


FIGURE 3.1: a) Probability of the distance between two randomly drawn configurations with length $n = 100$ and $n = 1000$ and with $k = 3$ and $k = 10$ colours. The partition distance is found by randomly sampling 10^6 pairs of configurations. b) The averaged partition distance between random configurations for different n and k.

running a heuristic search algorithm to find the best colouring. Although the heuristic is not guaranteed to find the optimum, by running it many times we can be fairly certain that the optimum has been found (see section 6.2.3 for a justification of this assertion).

Figure 3.2-b shows the proportion of the colourable graphs as a function of n and k. We observe two clear regions divided by a fairly steep transition region. This change in behaviour marks a "phase-transition", which is typical of many constraint satisfaction problems. As we increase the number of constraints, these problems undergo a transition from a satisfiable phase to an unsatisfiable phase. The sharpness of the transition typically grows with the instance size. In the context of graph-colouring, each edge represents a constraint. For a fixed number of vertices and colours, graphs are colourable with a small enough number of edges. As we increase the

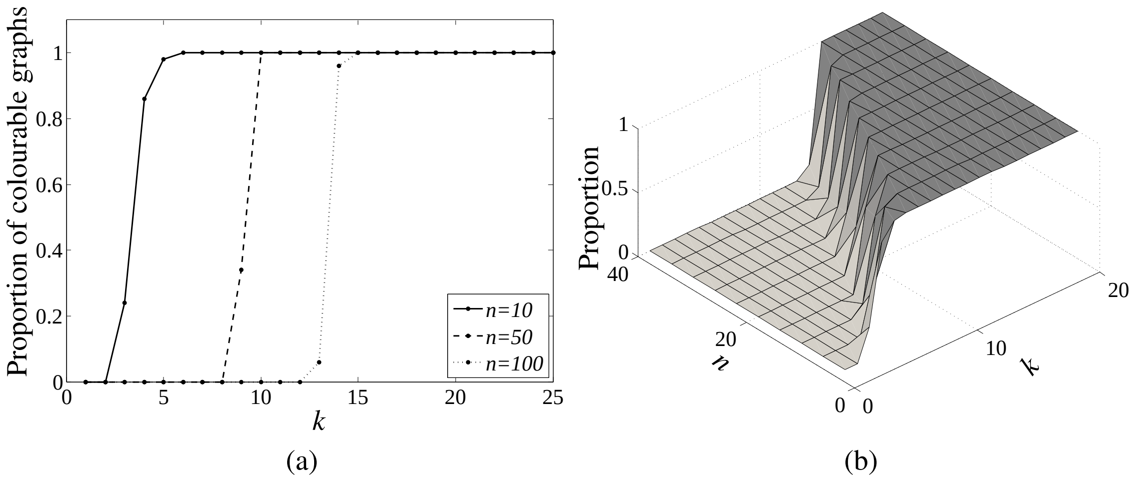


FIGURE 3.2: a) The proportion of colourable graphs for different values of problem size n and number of colours k for $p = 0.5$. The data are averaged over 1000 random problem instances. b) The proportion of colourable graphs as a function of system size n and number of colours k. The data are averaged over 1000 random problem instances.

number of edges, there are fewer ways to colour the graph. As we further increase the number of edges there comes a point where it is no longer possible to colour the graph without introducing a colour conflict. It is more common in graph-colouring to consider the graph (i.e., the number of constraints) as fixed and consider varying the number of colours. In this case, the unsatisfiable phase corresponds to the case of few colours, while the satisfiable phase corresponds to the case where we have many colours.

For an ensemble of graphs the transition between colourable and uncolourable graphs is less well defined, although we could define the location of the transition to be the point where at least half the graphs are colourable. The location of the phase-transition as a function of n, p and k is unknown, although a lower-bound on the probability of a graph drawn from $\mathscr{G}(n,p)$ being colourable is given by,

$$\sum_{\boldsymbol{n}\in\Lambda(n,k)} n! \prod_{i=1}^{k} \frac{(1-p)^{n_i(n_i-1)/2}}{n_i!},$$

where $\boldsymbol{n} = (n_1, n_2, \ldots, n_k)$ and $\Lambda(n,k)$ is the set of vectors, $\boldsymbol{n}$, such that $\sum_{i=1}^{k} n_i = n$ and $n_i \geq 0$ (Prügel-Bennett). In Figure 3.3, we show the lower-bound on the chromatic number versus n. For more information on the chromatic number of random graph see ball1988,Krivelevich98. We also show the minimum number of colours needed to colour random graphs drawn from $\mathscr{G}(n, 0.5)$, found by a modified version of hybrid-GA, which is currently the state-of-the-art for solving dense random graph-colouring problems (Galinier and Hao, 1999).

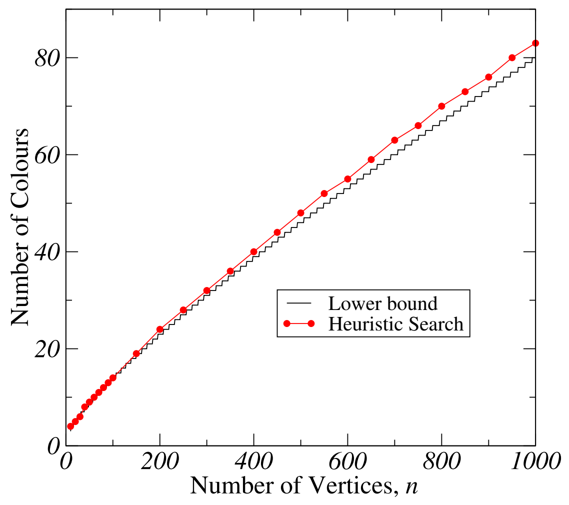


FIGURE 3.3: Lower-bound on the chromatic number for graphs drawn from $\mathscr{G}(n,0.5)$. Also shown are the best-colourings found by a state-of-the-art heuristic algorithm.

3.4 Methodology

As discussed in section 3.2, a feature of graph-colouring is that a permutation of the colour labels leads to the same partitioning of the vertices and thus has an identical cost. Thus with k colours there is a $k!$-fold permutation symmetry. In our analysis of local optima we will treat all configurations that lead to the same partitioning as equivalent. In practice, we accomplish this by permuting the colours in a solution to a canonical form (which is easily achieved by making the colour of vertex 1 equal to the first colour, then the next vertex that is coloured differently from vertex 1 we assign the second colour, etc.). Rather than store all the configurations for a partition, we just store the least configuration according to some ordering. This allows us to quickly check if we have visited a local optimum previously.

3.5 Landscape Analysis

In this section we study some properties of the fitness landscape of the graph-colouring problem, and show the effect of the size of the problem on the properties of the landscape.

3.5.1 Density of Configurations

We start by considering the density of configurations as a function of the cost. We also give formulae for the cumulants, which, as far as we know, have not been given elsewhere.

The expected cost of a graph, G with k colours is given by,

$$\bar{c}(G) = \mathbb{E}\left(c(G,\boldsymbol{x})\right) = \mathbb{E}\left(\sum_{(i,j)\in\mathscr{E}} [\![x_i = x_j]\!]\right) = \sum_{(i,j)\in\mathscr{E}} \mathbb{E}\left([\![x_i = x_j]\!]\right) = \frac{|\mathscr{E}|}{k},$$

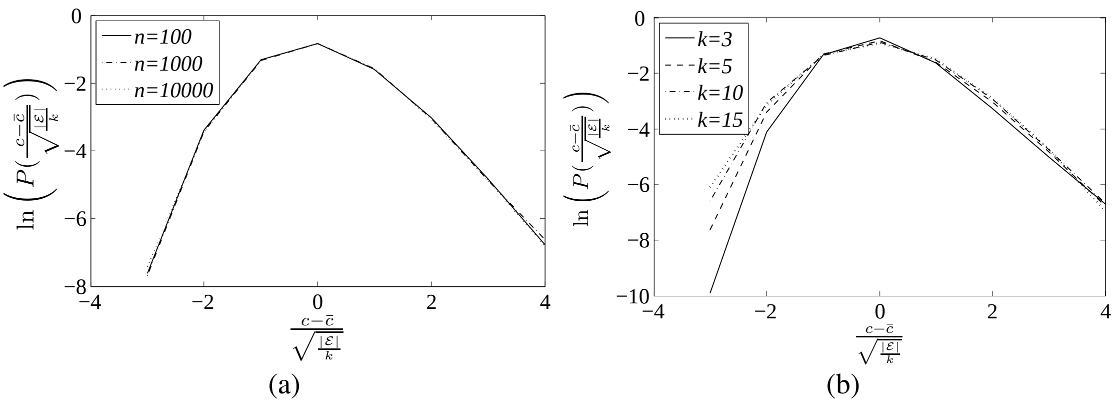


FIGURE 3.4: a) Natural logarithm of histogram of costs for random configurations in particular instances with $k = 5$ and $p = 0.5$ for $n = 100, 1000, 10000$. b) Natural logarithm of histogram of costs for random configurations in particular instances with $n = 100$ and k =3, 5, 10 and 15.

where k is the number of colours and $|\mathscr{E}|$ is the number of edges in the graph. For graphs drawn from $\mathscr{G}(n, p)$,

$$|\mathscr{E}| = p\frac{n(n-1)}{2}.$$

To find the density of configurations, we generate random colourings and compute a histogram of the costs. Figure 3.4-a shows the natural logarithm of the probability of a cost around the average cost scaled by $\sqrt{|\mathscr{E}|/k}$ for $k = 5$ and for $n = 100$, 1000 and 10 000. Regardless of the size of the problem, the results are almost similar for each randomly drawn problem instance. The cost of the random configurations are approximately bell shaped around the average cost with a variance of $\Omega(n^2)$. We observe that the distributions are quite skewed. The distribution hardly varies for different n. Figure 3.4-b shows the same distribution but now for fixed $n = 100$, and for different values of k. This shows that the distribution varies with the number of colours k.

3.5.2 Cumulants

To get a better understanding of the density of configurations, we can compute the moments or cumulants of the distribution. The i-th central moment is defined as,

$$\mu_i(G) = \mathbb{E}\left((c(G, \boldsymbol{x}) - \bar{c}(G))^i\right).$$

The cumulants are more natural measures of the distribution. They are closely related to the moments. The first cumulant is equal to the mean cost, and the second cumulant is equal to the second central moment or variance. The third cumulant is equal to the third central moment, while the fourth cumulant is equal to the fourth central moment minus three times the variance squared. (The cumulants arise as Taylor coefficients of the natural logarithm of the moment-generating function.) One property of the cumulants is that for a normal distribution, all the cumulants above the second cumulant are zero. Thus the cumulants can be viewed as a measure of the deviation from a normal distribution. To make these measures invariant of the cost scale, it is common to scale the n-th cumulant by $K_2^{n/2}$ (rescaling the cost would leave these scaled cumulants unchanged). The scaled third and fourth cumulants are known as the skewness and kurtosis,

$$\gamma_1 = \frac{K_3}{K_2^{3/2}}, \qquad\qquad \gamma_2 = \frac{K_4}{K_2^2}.$$

The skewness measures the asymmetry in the tails of a distribution, with a positive skewness indicating a longer tail on the right of the distribution and a negative skewness a longer tail on the left of the distribution. A positive kurtosis indicates that the tails are more pronounced than a Gaussian and a negative kurtosis indicates less probability mass in the tails.

To compute the cumulants we note that we can write the cost of a configuration (colouring) $\boldsymbol{x}$ for a graph G minus the average cost as,

$$c(G,\boldsymbol{x}) - \bar{c}(G) = \sum_{(i,j)\in\mathscr{E}} \left([\![x_i = x_j]\!] - \frac{1}{k} \right) = \sum_{e\in\mathscr{E}} S_e,$$

where $S_{(i,j)} = [\![x_i = x_j]\!] - 1/k$. On averaging over all colourings we find $\mathbb{E}\left(S_e\right) = 0$. Furthermore, if we have a set of edges, $\mathscr{F} \subset \mathscr{E}$, such that there exists a vertex that belongs to one and only one edge in $\mathscr{F}$, then $\mathbb{E}\left(\Pi_{e\in\mathscr{F}} S_e\right) = 0$. Thus, the variances in the cost for a graph G are given by,

$$\begin{aligned} K_2(G) &= \mathbb{E}\left((c(G,\boldsymbol{x}) - \bar{c}(G))^2 \right) = \mathbb{E}\left(\left(\sum_{e\in\mathscr{E}} S_e \right)^2 \right) \\ &= \mathbb{E}\left(\sum_{e\in\mathscr{E}} S_e^2 \right) + \mathbb{E}\left(\sum_{e\in\mathscr{E}} \sum_{e'\in\mathscr{E} e'\neq e} S_e S_{e'} \right) = \sum_{e\in\mathscr{E}} \mathbb{E}\left(S_e^2\right) = |\mathscr{E}|\,\mathbb{E}\left(S_e^2\right). \end{aligned}$$

We can straightforwardly evaluate $\mathbb{E}\left(S_e^2\right)$,

$$\begin{aligned} \mathbb{E}\left(S_{(i,j)}^2\right) &= \mathbb{E}\left(\left([\![x_i = x_j]\!] - \frac{1}{k} \right)^2 \right) = \mathbb{E}\left([\![x_i = x_j]\!] - \frac{2}{k} [\![x_i = x_j]\!] + \frac{1}{k^2} \right) \\ &= \frac{1}{k} - \frac{2}{k^2} + \frac{1}{k^2} = \frac{1}{k}\left(1 - \frac{1}{k}\right), \end{aligned}$$

where we have used the fact that because $[\![x_i = x_j]\!] \in \{0,1\}$ then $[\![x_i = x_j]\!]^2 = [\![x_i = x_j]\!]$. Thus, the variance is given by $K_2 = |\mathscr{E}|\,(k-1)/k^2$. For a graph drawn from the ensemble $\mathscr{G}(n,p)$, the expected number of edges is $p\,n\,(n-1)/2$. Thus, the expected variances for a graph drawn from $\mathscr{G}(n,p)$ is,

$$K_2 = \frac{p\,n\,(n-1)\,(k-1)}{2k^2}.$$

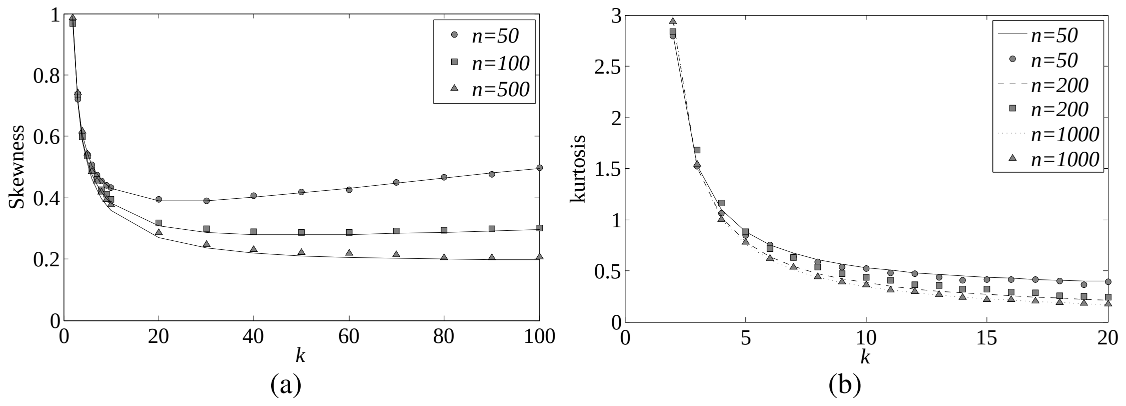


FIGURE 3.5: a) The skewness of the distribution of the cost of the configurations for different n and k for $p = 0.5$. b) The kurtosis of the distribution of the cost of the configurations for different n and k for $p = 0.5$.

We can use the variance to define a measure of the cost diversity defined as,

$$v = \frac{\sqrt{K_2(G)}}{\bar{c}(G)} = \sqrt{\frac{k-1}{|\mathscr{E}|}}.$$

The higher-order cumulants can be found in a similar fashion to the second order cumulant. The details are given in appendix D. The expected third order cumulant is given by,

$$K_3 = \frac{pn(n-1)(k^2-3k+2)}{2k^3} + \frac{p^3 n(n-1)(n-2)}{k^3}.$$

Figure 3.5-a shows the analytical curve and the empirical skewness for different values of k and n for $p = 0.5$.

The expected fourth cumulant is given by,

$$K_4 = \frac{pn(n-1)(k-1)}{2k^4}(6-6k+k^2+48p^2-24kp^2-24np^2+12knp^2+24p^3-24np^3+6n^2p^3). \tag{3.3}$$

(See appendix D for details.) The analytical curve and the empirically measured results for kurtosis of the distribution, for different values of n and k are shown in Figure 3.5-b. The results show that as n and k grow, the kurtosis decreases gradually, although it is more sensitive to the number of colours than the size of the problem.

3.5.3 Auto-Correlation

For a graph, G, the auto-correlation is given by,

$$\mathscr{R}(\tau) = \frac{1}{K_2(G)}\mathbb{E}\Big(\left(c(G,\boldsymbol{x}(t+\tau)) - \bar{c}(G)\right)\left(c(G,\boldsymbol{x}(t)) - \bar{c}(G)\right)\Big),$$

where $\boldsymbol{x}(t)$ is the configuration at step t of a random walk, and $K_2(G)$ is the variance in the cost for random configurations of the problem instance, G. As each step is chosen independently, the correlation is reduced by the same factor at each step so that,

$$\mathscr{R}(\tau) = \mathscr{R}(1)^{\tau} = \mathrm{e}^{-\tau/l},$$

where $l = -1/\ln(\mathscr{R}(1))$ is the correlation length. We suppose that in one step we change the colouring of vertex i from x_i to x_i'. Denoting the neighbours of vertex i by $\mathscr{N}_i = \{j|(i,j) \in \mathscr{E}\}$ then,

$$\mathscr{R}(1) = \frac{1}{K_2(G)} \mathbb{E}\Big((c(G,\boldsymbol{x}(t+1)) - \bar{c}(G))\,(c(G,\boldsymbol{x}(t)) - \bar{c}(G)) \Big) = \frac{1}{K_2(G)} \left(K_2(G) + C(G)\right),$$

where we have separated out the term that differs from $K_2(G)$,

$$C(G) = \mathbb{E}\left(\sum_{j \in \mathscr{N}_i} \left([\![x_i' = x_j]\!] - [\![x_i = x_j]\!] \right) \left([\![x_i = x_j]\!] - \tfrac{1}{k} \right) \right).$$

Using the properties of indicator functions,

$$[\![x_i' = x_j]\!]\, [\![x_i = x_j]\!] = 0, \qquad\qquad [\![x_i = x_j]\!]^2 = [\![x_i = x_j]\!],$$

and also that on average $|\mathscr{N}_i|$ is equal to, $2|\mathscr{E}|/n$ we find

$$\mathscr{R}(1) = \frac{1}{K_2(G)} \left(K_2(G) - \frac{2|\mathscr{E}|}{nk} \right) = 1 - \frac{2k}{n(k-1)}$$

(recall that $K_2(G) = |\mathscr{E}|(k-1)/k^2$). Subsequently the correlation length is approximately equal to $n(1-1/k)/2$. The correlation length thus grows linearly with n and is smaller for smaller k. Larger problems thus appear smoother, although this just reflects the fact that changing the colour of a single vertex has a smaller relative change on the cost.

3.5.4 Time to Local Optimum

Having described the average behaviour of configurations, we now move on to study properties of local and global optima. Our analysis begins by studying the time taken by the local-search algorithm to reach a local optimum. First we will study the relationship between the time to reach a local optimum and the number of colours. There is a strong relationship between the number of steps taken by a local search algorithm to reach a local optimum and the number of colours. This relationship is illustrated in Figure 3.6-a, where the mean of the steps needed to reach a local optimum is shown plotted against k, for $n = 30$, 40, 50 and $k = 1, \ldots, 40$. The curves consist of a phase-transition and three major parts, 'A', 'B' and 'C' (See Figure 3.7-a). In the first part, for $k = 2$ up to $k = 9$ there is a rapid increase in the number of steps to a

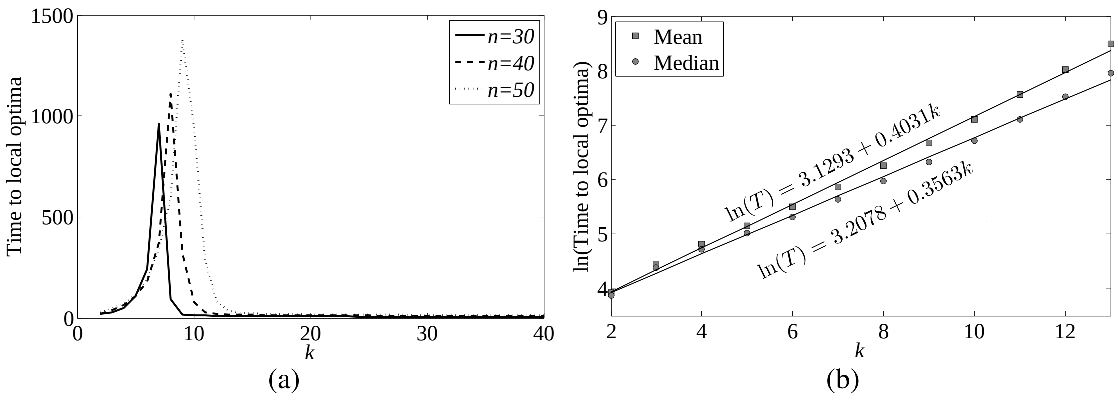


FIGURE 3.6: a) Time to reach a local optimum versus the number of colours for $n = 30$, 40 and 50. Each data point represents the mean over 100 instances and 10^5 hill-climbs per instance. b) Natural logarithm of the time to reach a local optimum versus the number of colours. The size of the problem is $n = 100$ and $k = 2$ up to chromatic number around $k = 13$. Each data point represents the mean or median over 100 instances and 10^5 hill-climbs per instance.

local optimum. For problems of this size the chromatic number is around 9 (see Figure 3.2-a)—this marks the phase transition between unsatisfiable and satisfiable instances. After the phase-transition, in part '*B*', there is a sharp decrease in the time to reach a local optimum. During part '*C*', the number of steps decays at a much slower rate. For all the instances we have studied, the phase-transition occurs at the chromatic number, i.e., the minimum k such that the global optima have a cost of zero.

At each part, the graph shows different behaviour. On a logarithmic scale, at part '*A*' the data fit a straight line, meaning that for k up to the chromatic number, the number of steps grows exponentially with the number of colours. This is shown in Figure 3.6-b, where on a logarithmic scale, the mean and median of the time to reach a local optimum are plotted against k for $n = 100$ and k up to the chromatic number. The practical data fit the straight line on the logarithmic scale; therefore the data suggest that the average time to local optima for $n = 100$ is

$$\overline{T} = 22.64 \times e^{0.40k},$$

for $k = 2$ up to the chromatic number. The reason for such behaviour is that up to the chromatic number as k grows, the size of the plateaux in the landscape increases exponentially, causing the time to local optima to increase exponentially. We note that the dramatic change in behaviour of the time to reach a local optimum is not reflected in the auto-correlation function. All that happens to the auto-correlation as we increase k is that the correlation length slowly increases. This demonstrates clearly that the time to reach a local optimum is a property of the large-scale behaviour of the landscape, and is unrelated to the local smoothness of the landscape.

The second and the third parts of the graph show different behaviours. The plot of the time to local optima as a function of k for $n = 50$ is shown in Figure 3.7-a on a log-log scale. By rescaling the graph, the data for the part '*B*' and '*C*' fit a straight line, suggesting that the time to

TABLE 3.1: Time to local optima as a function of k for different n. The results are averaged over 100 instances each 10^4 hill-climbs.

n	T_A	T_B	T_C
30	$2.84 \times e^{0.77k}$	$\propto k^{-16.08}$	$\propto k^{-0.62}$
40	$5.53 \times e^{0.62k}$	$\propto k^{-11.80}$	$\propto k^{-0.61}$
50	$8.14 \times e^{0.55k}$	$\propto k^{-12.94}$	$\propto k^{-0.60}$
60	$10.72 \times e^{0.50k}$	$\propto k^{-14.56}$	$\propto k^{-0.59}$
70	$13.94 \times e^{0.46k}$	$\propto k^{-14.42}$	$\propto k^{-0.59}$
80	$14.77 \times e^{0.46k}$	$\propto k^{-18.76}$	$\propto k^{-0.57}$
90	$15.73 \times e^{0.46k}$	$\propto k^{-20.31}$	$\propto k^{-0.54}$

local optima decays as a factor of k^g, where $g < 0$ is the gradient of the fitting line. The fitting curve for part 'B' is $T \propto k^{-14.76}$ and for the part 'C' is $T \propto k^{-0.62}$.

The same experiment is performed on different sizes of the problem and the formulae describing the behaviour of the time to local optima as a function of k are summarised in Table 3.1. For the part 'A' (k up to the chromatic number), as n grows, the rate of growth reduces, although the constant increases so that for a fixed k, the expected number of steps to reach a minimum increases slowly with n in this regime. However, as the chromatic number increases with n, the expected number of steps to reach a minimum increases substantially with n for k at the chromatic number. The interesting behaviour of the part 'B' is that for $n = 30$ up to 100, it covers just 3 or 4 numbers of colours, i.e., $k = \chi(n)$ to $k = \chi(n)$+3. This is why in Table 3.1 the decay rate of the time to local optima grows with the system size; for bigger n it has to fall off from a larger number within just three or four steps. In part 'C' the decay rate decreases with the system size.

The time to local optima also depends on the system size, n. Figure 3.7-b shows the mean and median time to reach a local optimum versus the problem size n for $k = 4$. The graph indicates that below the phase-transition, the time apparently increases linearly with n. The best straight line fit to the data are also shown in this Figure. The same method is used to find the time to local optima as a function of n for different k, and the results are shown in Table 3.2. The Table is divided into three parts, below the chromatic number, where the number of steps grows linearly with n, at the chromatic number, where it grows exponentially with n, and above the chromatic number, where the time grows polynomially with n. Using this extrapolation, the mean time a local-search algorithm takes to reach a local optimum for a problem with size of $n = 1000$, and $k = 4$ is 1130. Although some caution is needed in extrapolating from small instances to large instances, nevertheless, for a small number of colours a local-search algorithm does not need much time to find a local optimum.

As we saw, the time to reach a local optimum depends on both the size of the system and the number of colours. In Figure 3.8, the log time to local optima is plotted against k and n. The three regions are seen in this figure, and the relationship between the size of the system and the time to local optima is different in each region. Note that at the phase transition on a logarithmic

TABLE 3.2: Time to local optima as a function of n for different k. The results are averaged over 100 instances, each 10^4 hill-climbs. This is for $n = 30, 40, ..., 90$.

k	T_{Mean}	T_{Median}
2	$3.26 + 0.46n$	$1.27 + 0.46n$
3	$4.53 + 0.82n$	$0.64 + 0.80n$
4	$17.44 + 1.11n$	$5.76 + 1.06n$
5	$53.90 + 1.18n$	$16.48 + 1.35n$
$\chi(n)$	$114.48 \times e^{0.053n}$	$29.52 \times e^{0.059n}$
30	$-3.45 + n^{1.56}$	$-3.55 + n^{1.58}$
40	$17.44 + n^{1.59}$	$-3.88 + n^{1.62}$
50	$53.90 + n^{1.60}$	$-4.29 + n^{1.68}$

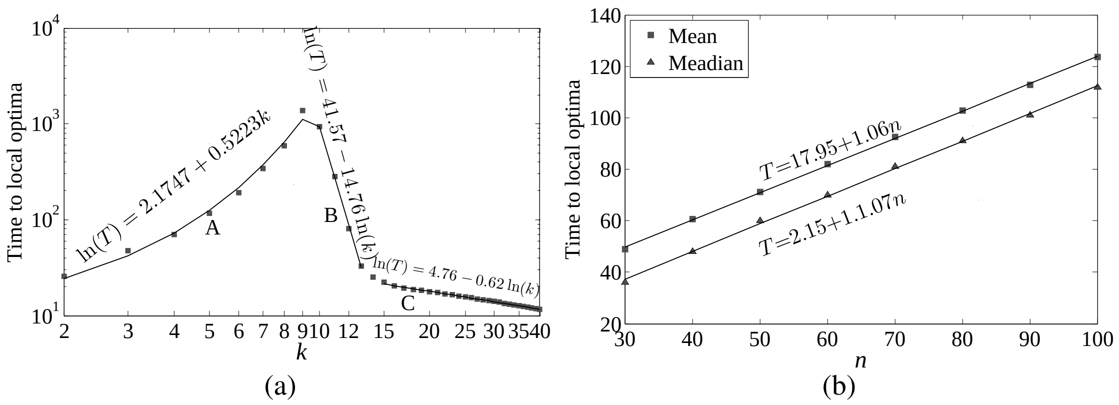


FIGURE 3.7: a) Time to reach a local optimum versus the number of colours k, on a log-log plot. The size of the problem is $n = 50$. Each data point represents the mean over 100 instances and 10^5 hill-climbs per instance. b) Time to reach a local optimum versus the problem size. The number of colours is $k = 4$. Each data point represents the mean or median over 100 instances and 10^5 hill-climbs per instance.

scale the time grows linearly with n, meaning the time increases exponentially with the system size. The phase transition in Figure 3.2-b, which occurs at the chromatic number, is also shown in Figure 3.8 with a dashed line. The graph clearly shows that the peak in the time to local optima coincides with the chromatic number.

It is relatively easy to understand this intuitively. If we consider a random colouring, then the probability of a colour conflict at any edge is $1/k$. Consider a vertex, i with $|\mathcal{N}_i|$ neighbours. On average, $|\mathcal{N}_i|$ is equal to $p(n-1)$. The probability that c_i of these edges are unsatisfied is given by a binomial distribution,

$$\mathbb{P}(c_i) = \binom{|\mathcal{N}_i|}{c_i} \left(\frac{1}{k}\right)^{c_i} \left(1 - \frac{1}{k}\right)^{|\mathcal{N}_i| - c_i}.$$

We can think of c_i as the local colour cost for vertex i. The total cost is equal to half the sum of the local colour cost (the half occurs because we double count each edge). The distribution of local colour costs are shown schematically in Figure 3.9 for $k_1 \ll \chi \ll k_2$.

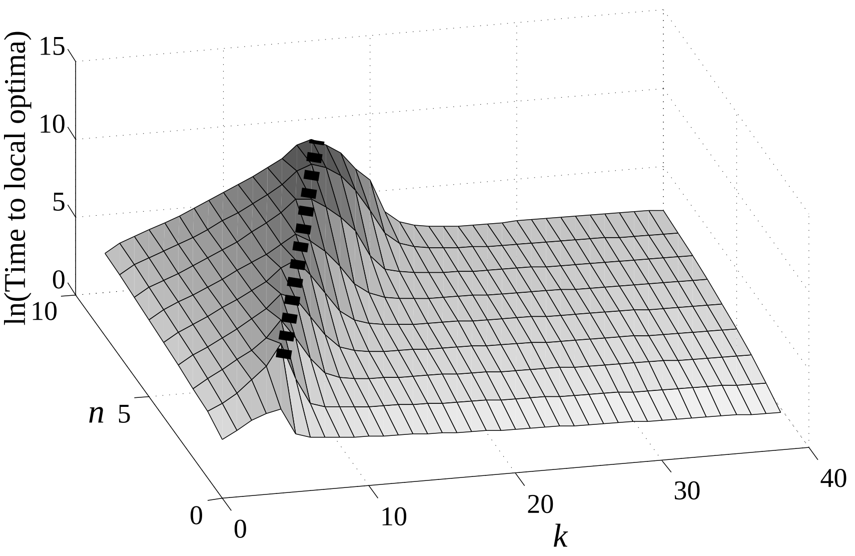


FIGURE 3.8: Time to reach a local optimum versus the problem size. The number of colours is $k = 4$. Each data point represents the mean or median over 100 instances and 10^5 hill-climbs per instance.

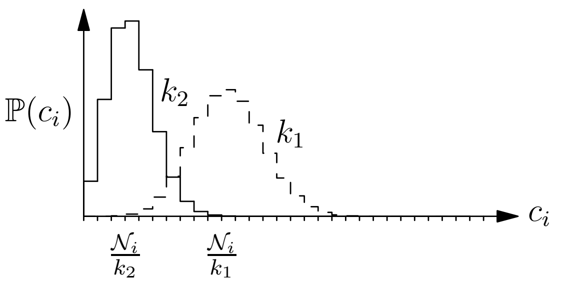


FIGURE 3.9: Schematic of the probability distribution of the number of unsatisfied edges connected to an arbitrarily chosen edge.

The probability of a vertex, i, having no colour conflicts is thus,

$$\mathbb{P}(c_i = 0) = \left(1 - \frac{1}{k}\right)^{|\mathcal{N}_i|}.$$

If k is well above the phase-transition, then it will be highly probably that we can choose a colour for vertex i, so that it has no colour conflicts. If we do this for all vertices, then we will have found a configuration with zero cost. Furthermore, typically there will be some fraction of the vertices where we have an option of several colours to choose, all of which give no colour conflicts. As a result, we are likely to have an exponentially large global optimum (e.g. if some proportion, f, of vertices have zero local cost for two colours, we might expect around 2^{fn} configurations in the global optimum—of course, by choosing a vertex colour we modify the neighbourhood for the other vertices, so this argument should not be taken too literally). In this regime we can quickly find a solution with cost 0, at which point we can stop our search.

As we decrease k (or increase the probability of an edge), then $\mathbb{P}(c_i = 0)$ decreases and it

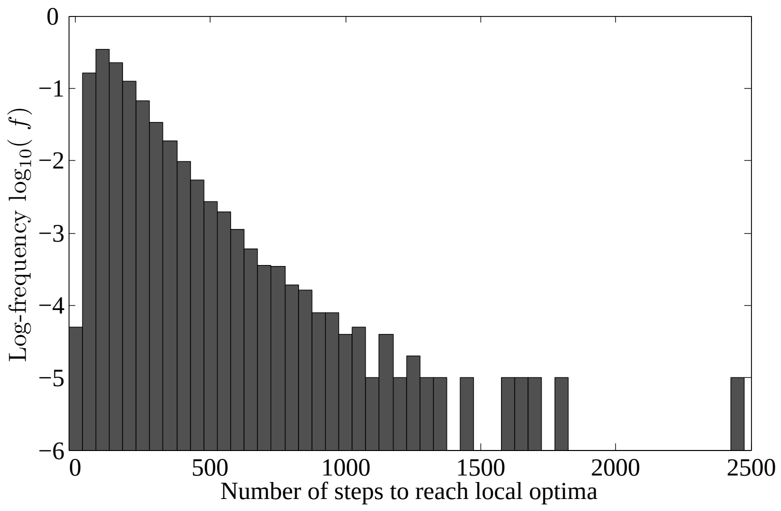


FIGURE 3.10: Logarithm of the histogram of the number of steps to reach the local optima for a particular instance for $k = 5$ and $n = 100$. The data are collected on 10^5 hill-climbs.

becomes harder to find a colour for a vertex that does not produce a colour conflict. Around the phase-transition we would expect that we can (at least, after some search) find a colouring for most vertices with local colour cost zero. Furthermore, there are likely to be multiple alternatives for some fraction of the vertices leading to an exponentially large connected set at the same cost. As we cannot stop our search unless we have found a configuration of cost 0, our search is likely to take exponentially longer to exhaustively search all neighbours at a low cost.

For a smaller number of colours (e.g. k_1 in Figure 3.9), it becomes unlikely that we can find a colour for most vertices with no conflicting edges. Thus, a local search algorithm will find a colour such that the probability of the cost, $\mathbb{P}(c_i)$, is very small. Because of this, it is unlikely for there to be alternative colours with the same cost c_i. (This is in direct contrast to the case for large k. In that case, many vertices will have a local colour cost of 0 and because it is impossible to have a lower local colour cost, there are likely to be many neighbouring configurations with zero local colour cost at the same vertices.) Thus, in this regime the optima tend to be relatively small, and the time to reach the local optima short, as the search does not spend a long time on a plateau. However, as we will see, another difficulty arises for these instances; that is, there tends to be an exponential number of local optima.

As observed in Figure 3.6-a, the mean time to reach a local optimum is higher than the median, indicating that the distribution of steps to reach a local optimum has a long tail. Up to the phase-transition, the gap between the mean and the median increases as k grows, meaning that for larger k, the tail of the distribution becomes longer. Figure 3.10 shows the distribution of times to reach a local optimum plotted on a semi-log scale to show the rare events. These data were gathered on a particular randomly constructed problem instance. The data show that occasionally it takes a very long time for a hill-climbing algorithm to reach a local optimum.

The analysis presented so far does not show the size of the plateaus and the height of the cliffs the algorithm sees at each stage of the search process. In order to study this, a record of the steps

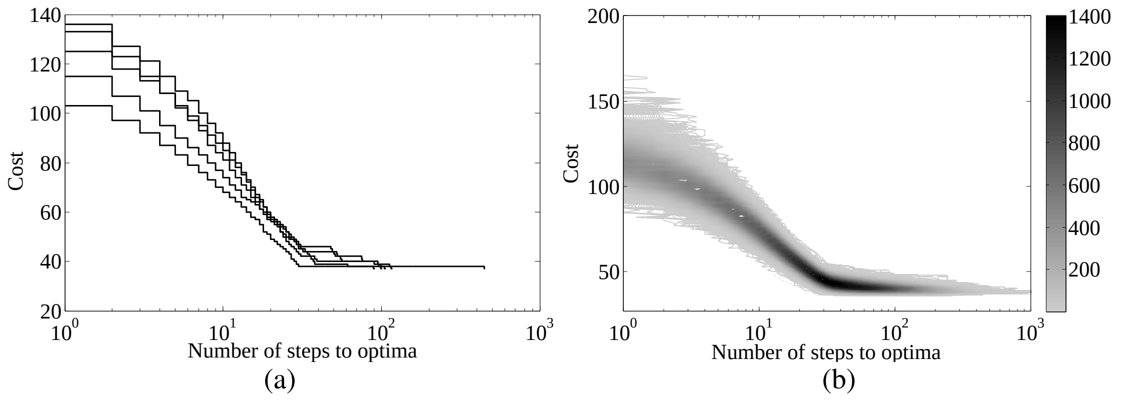


FIGURE 3.11: a) The record of the steps taken by the local-search algorithm to get to the most visited global optimum, for five different search processes starting from random configurations. The cost of the configurations at each step is represented against the number of steps plotted on a logarithmic scale. The size of the problem is $n = 50$ and the number of colours is $k = 5$. b) The density plot of the record of the steps taken by the local search algorithm to get to the local optima for 10^5 different search process, starting from random configurations. The cost of the configurations at each step is represented against the natural logarithm number of steps. The size of the problem is $n = 50$ and the number of colours is $k = 5$.

taken by the local-search algorithm to get to the most visited global optimum, for five different descents from random configurations, for $n = 50$ and $k = 5$, is shown in Figure 3.11-a. The graph shows the cost of the solutions at each step against the natural logarithm of the number of steps. We observe that the shape of the landscape changes as the local-search algorithm gets closer to the local optima. At the beginning, when the search algorithm starts from a random configuration, the algorithm easily finds a lower cost neighbour. Furthermore, the algorithm initially makes a large improvement in cost at each step (note that the algorithm always chooses the best neighbour to move to). However, as the search progresses, not only does the probability of finding a better configuration decrease, but the improvement in cost also decreases. Note that we stop the local-search algorithm when it reaches a local optimum. Thus, the number of steps the algorithm takes on a local optimum to make sure it has reached one, is not counted in Figure 3.11-a.

In order to show the general behaviour of the record of the steps to reach a local optimum, a density graph of the record of the steps taken by the local-search algorithm to get to the local optima, for 10^5 different search process starting from random configurations, is shown in Figure 3.11-b. The horizontal axis shows the natural logarithm of the number of steps to reach a local optimum. Most runs reach a local optimum in 30 to 100 steps, but in some rare cases it takes many more steps to get to local optima. Although showing different behaviour for different search processes, all the descents show similar shape. At the beginning of the search, it is easy to find a better solution, but getting closer to the local optima, the probability of a neighbour being better becomes exponentially smaller.

The graph can be divided in two different parts. In the first part all the runs make steady progress, although the probability of an improving move decreases exponentially as the optimum is approached. The second part (starting after around 50 steps in this case) is due to a small number of runs, where a very large plateau is reached with a cost just above the optimum. When this happens, it can take over an order of magnitude longer to search this plateau and find the optimum.

The time it takes to find a local optimum can be a problem when the number of colours is close to the chromatic number. Another feature, however, which makes graph-colouring hard for local search is the sheer number of local optima. This is a problem both around the chromatic number and in the over-constrained region (when k is below the chromatic number). We explore this next.

3.5.5 Number of Local Optima

To find the number of local optima in the landscape we use the exhaustive local-search algorithm and store the local optimum it returns. As mentioned earlier, we treat every local optimum obtained by permuting the colours as a single local optimum. Therefore because of the symmetry in the landscape, there are $k!$ colour permutations of a local optimum. Before storing the local optima we break the symmetry; thus there are in fact $k!$ times more local optima in the colour space than reported.

The mean, minimum and maximum number of times the local optima at certain cost were found for a particular problem instance with $n = 50$ and $k = 7$ are shown in Figure 3.12. Of course there are likely to be local optima that we have not found. However, note that for costs smaller than 12, we have visited each local optimum at least 30 times. If any local optima exist with a cost less than 12 that we have not observed, then they would appear to have unusually small basins of attraction.

Fitting the data for the mean count of the local optima at each cost level in Figure 3.12 to a straight line gives a way to estimate the probability of finding a local optima. For the case of Figure 3.12, where $n = 50$ and $k = 7$, the gradient of the fitting line on the data for cost $c = 10$, $\ldots$, 14, is -0.67, which suggests that the probability of getting to a particular local optimum at the cost c, decays exponentially as $P_v(c) \propto 10^{-0.67c}$. (Note that because there are many more local minima at high cost, this does not imply that the search is likely to reach the global optimum). We can give no strong justification for believing that this exponential decay should continue for high costs, although it seems to be a fairly good approximation at low cost for every instance we have examined.

A faster decay rate means that the inferior local optima have much smaller basins of attraction and the local-search algorithm has a better chance of reaching a fit local optimum. The decay rate of the probability of finding a local optimum at cost c for different sizes of the problem and different number of colours is summarised in Table 3.3. The data are averaged over 100 problem instances and 10^5 number of hill climbs. Due to practical reasons, we have made the data for k up to the chromatic number. The data suggest a linear relationship between the decay rate with both the system size and the number of colours. As the system size grows the decay

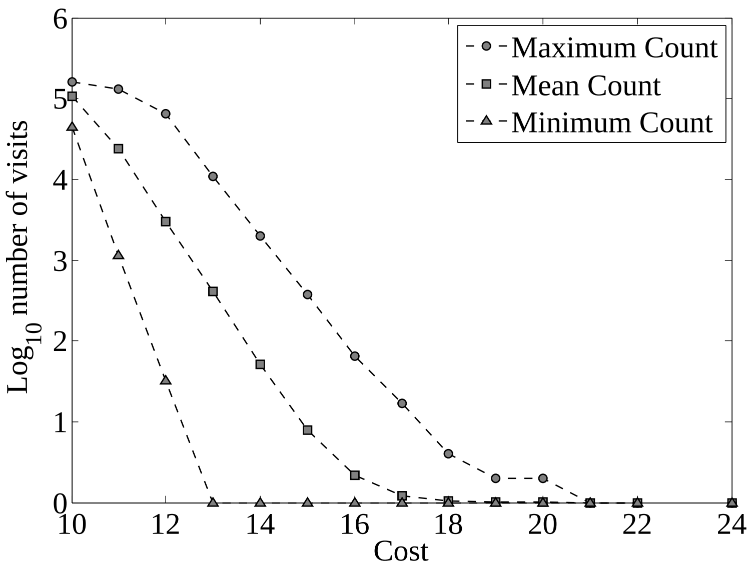


FIGURE 3.12: Logarithm of the number of times local optima at certain cost are hit in an experiment with 10^7 hill-climbs for a particular randomly drawn instance with $n = 50$ and $k = 7$.

TABLE 3.3: Probability of finding global optima at cost c as a function of n for different values of k is equal to 10^{-xc}, where x for different values of n and k are shown in the table.

	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=11	k=12
n=20	0.27	0.49	0.69	-	-	-	-	-	-	-	-
n=30	0.24	0.42	0.60	0.68	-	-	-	-	-	-	-
n=40	0.24	0.33	0.44	0.56	0.72	0.80	-	-	-	-	-
n=50	0.21	0.30	0.40	0.48	0.54	0.67	0.72	-	-	-	-
n=60	0.22	0.30	0.35	0.40	0.44	0.53	0.55	0.56	-	-	-
n=70	0.20	0.26	0.31	0.34	0.39	0.41	0.44	0.47	0.40	-	-
n=80	0.22	0.26	0.28	0.31	0.33	0.33	0.31	0.30	0.30	0.37	-
n=90	0.21	0.23	0.24	0.20	0.22	0.21	0.20	0.17	0.17	0.13	0.10
n=100	0.21	0.24	0.25	0.18	0.14	0.13	0.12	0.11	0.08	0.08	0.07

rate decreases linearly. This implies that as the system size grows, the probability of a local-search algorithm getting to a fitter local optimum shrinks exponentially. For larger systems, an inferior local optimum has relatively greater chance of being found. In the case of the number of colours, there is a positive linear relationship between k and the decay rate. As the number of colours increases, the chance of the local-search algorithm finding a particular fit local optimum increases. Interestingly, such behaviour is seen for n up to 80; however, for larger systems, an increase to the number of colours leads to a decrease in the decay rate. This means that for larger systems, as we get closer to the chromatic number, a local search algorithm has a smaller chance of finding a fit local optimum.

The relationship between the size of the problem and number of local optima is shown in Figure 3.13-a, where the logarithm of the number of observed local optima divided by $\sqrt{n}$ is plotted versus the cost divided by the average cost, $\bar{c} = |\mathscr{E}|/k$. The results are averaged over 100 instances of size 30, 45, 60 and 75. We have performed 10^5 hill-climbs to find the local optima. For $n = 60$ and $n = 75$ we significantly underestimate the actual number of local optima at higher costs (inferior local optima). Since the number of colours for all the problem sizes is taken to be $k = 5$, as n grows, the cost grows and the graphs are shifted to the right. Furthermore, the plot shows that this curve is roughly similar for different n. This indicates that the number of local optima grows approximately exponentially with the square root of the number of the vertices.

The number of local optima also depends on the number of colours. This is shown in Figure 3.13-b, where the mean number of local optima found in 10^5 hill-climbs, averaged over 100 instances is plotted against k. Since the time to reach a local optimum grows very rapidly close to the chromatic number (see section 6.2.1), it is difficult to count the number of local optima for large k. In Figure 3.13-b, we could make the data for k up to 8. The reason that such a limitation for the number of colours did not exist in counting the number of steps to local optima but exists in finding the number of local optima is that in finding the number of steps we can stop the search process as soon as we reach a solution with cost zero (a solution with the cost zero is clearly at a global optimum). In counting the number of local optima we have to explore all the solutions on the local optima to be able to label the optima. For k slightly above the chromatic number, the number of solutions on the local optima is huge, such that it is impossible to store all of them.

The relationship between the number of local optima and k is more complicated than its relationship with n. At first as k grows the number of local optima increases, but then it starts to gradually fall off. Although for k greater than a particular number, the number of local optima does not increase, that does not mean that finding a global optimum does not become harder for bigger k. Note that in Figure 3.13-b we count all local optima that are equivalent up to a colour permutation as a single optimum. The total number of Hamming optima is thus $k!$ greater than that shown in the Figure. We also observe that, although the number of local optima may not increase (from the point of view of partition distance), up to the chromatic number, the number of steps needed to reach a local optimum increases exponentially, making it harder to find them.

Figure 3.14 shows the proportion of local optima at a particular cost, and the probability of finding a local optimum with the local-search algorithm for a particular problem instance with $n = 50$ and $k = 5$. Note that Figure 3.14 shows the empirically measured proportion of local optima at each cost level, and thus underestimates the true proportion of local optima at high cost.

3.5.6 Reaching the Global Optima

We showed for the MAX-SAT problem that the gap between the expected cost found and the cost of the globally optimal solutions grows with the system size, n. This is also the case for the

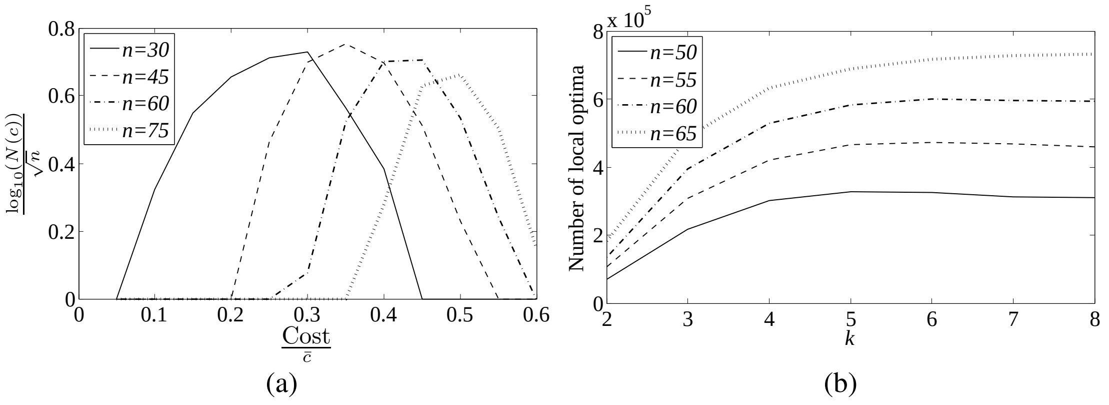


FIGURE 3.13: a) Logarithm of the number of times local optima at certain cost for different n and $k = 5$. The results are averaged over 100 instances for 10^5 hill-climbs. b) Mean number of local optima found in 10^5 hill-climbs averaged over 100 instances against the number of colours k.

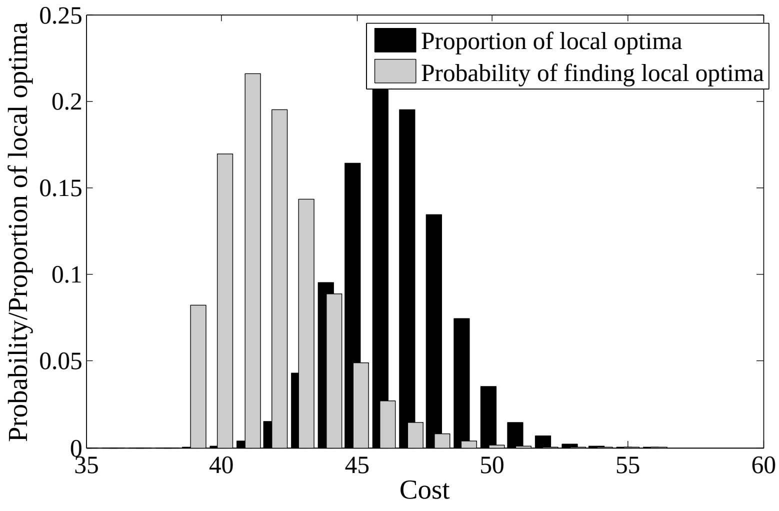


FIGURE 3.14: Histogram showing the proportion of local optima at a particular cost, and the probability of finding a local optimum of a particular cost for one instance with $n = 50$ and $k = 5$.

Graph-Colouring problem and is illustrated in Figure 3.15-a, where the minimum cost (average cost of the global optima over 500 problem instances) and the mean cost of the local optima found by the local-search algorithm over an ensemble of randomly drawn instances are shown. We observe empirically that the minimum cost is approximately consistent with,

$$c_{min} \approx \left(1 - \frac{5}{\sqrt{n}}\right)\bar{c}, \tag{3.4}$$

where $\bar{c}$ is the average cost of a configuration. The $\sqrt{n}$ scaling is not obvious (certainly as far as the authors are concerned). We note that the gap in Figure 3.15-a appears to be fairly constant with n. As the y-axis is scaled by the average cost, $\bar{c} = |\mathscr{E}|/k = O(n^2)$, the gap between the expected cost and minimum cost appears to increase with the square of the system size (i.e., proportional to $\bar{c}$).

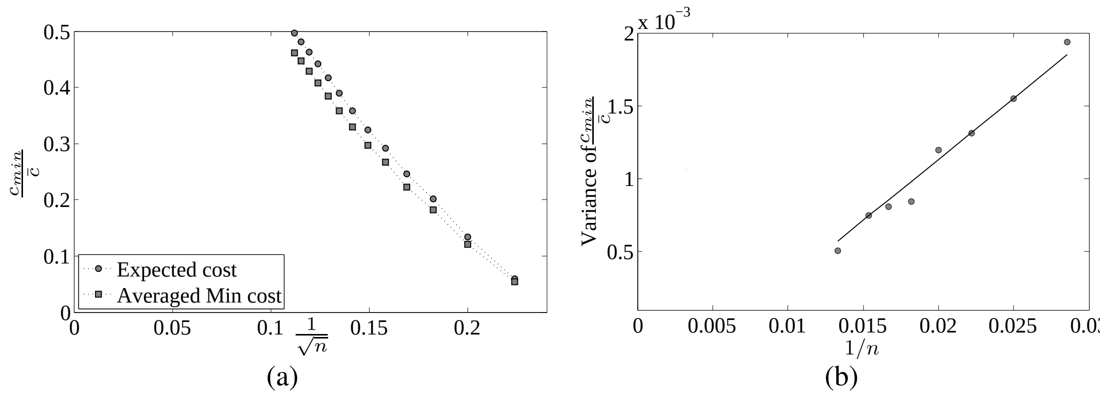


FIGURE 3.15: a) Plot of the expected minimum cost and the average cost found by local-search algorithm versus $1/\sqrt{n}$ for $k = 5$. The number of hill-climbs is 10^5 on 500 randomly generated instances. b) Plot of the variance of $c_{min}/\bar{c}$ found by local search algorithm. The number of hill-climbs is 10^5 on 500 randomly generated instances.

For a fixed instance, the cost of the optima found by the exhaustive search algorithm fluctuates on every run. Figure 3.15-b shows the variance in the cost found by different runs of local search divided by $\bar{c}$, plotted against $1/n$ (although the data are consistent with a straight line fit, we know that the variance cannot become negative, so must fall off somewhat slower for large n). Since the gap between global minimum cost and the cost found by local search does not decrease with n, then as n grows, finding a global optimum becomes more and more unlikely.

The gap between the expected cost and the cost of the global optima also depends on the number of colours, k. Figure 3.16-a shows the expected cost and the minimum cost of the local optima for $n = 50$, plotted against $\sqrt{k}$. Based on the scaling, it is clear that as the number of colours grows, the gap shrinks. It is easy to understand why this happens: the greater number of colours means that, on average, the local optima have smaller costs.

In Figure 3.16-b, the logarithm of the probability of finding a global optimum for $k = 5$ is plotted against n. The straight line fit is consistent with the hypothesis that finding a global optimum using the local-search algorithm becomes exponentially unlikely as the system size grows. Using the straight line fit in Figure 3.16-a to extrapolate to large n, the probability of the local-search algorithm finding a global optimum for an instance of size 1000 would be 1.24×10^{-36}. Employing the same method used in Figure 3.16-b, the probability of finding the global optima as a function of n is found for different k, and is presented in Table 3.4. The data show that for all the values of k, the chance of the local-search algorithms finding a global optimum decreases exponentially. Interestingly, the decay rate grows for larger k.

3.5.7 Number of Global Optima

Figure 3.17-a shows a histogram of the number of global optima for 10 000 random graph-colouring instances drawn from $\mathscr{G}(50, 1/2)$ and $k = 5$. We observe that there is quite a large spread (from instance to instance) in the number of global optima.

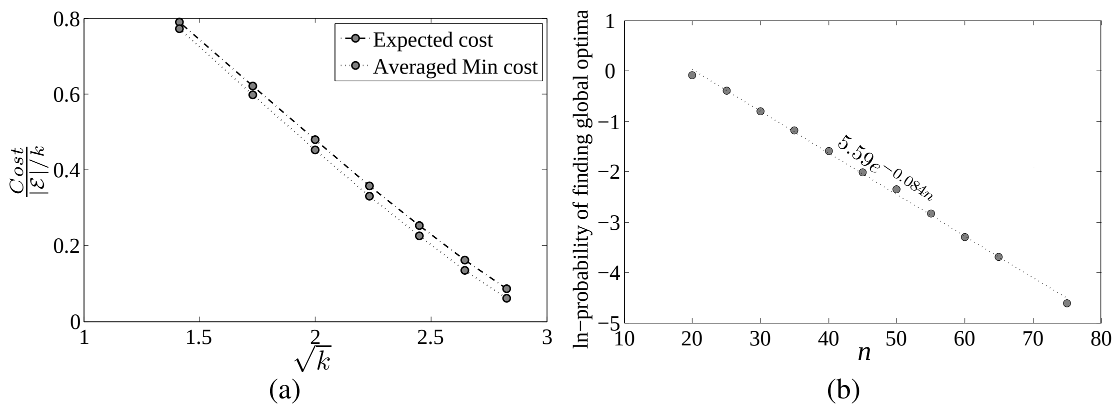


FIGURE 3.16: a) Plot of the expected minimum cost and the average cost found by local-search algorithm versus $\sqrt{k}$ for $n = 50$. The number of hill-climbs is 10^5 on 500 randomly generated instances. b) Natural log-probability of finding the global optima versus n for $k = 5$, the number of hill-climbs is 10^5 and the graphs are averaged over 500 randomly generated instances.

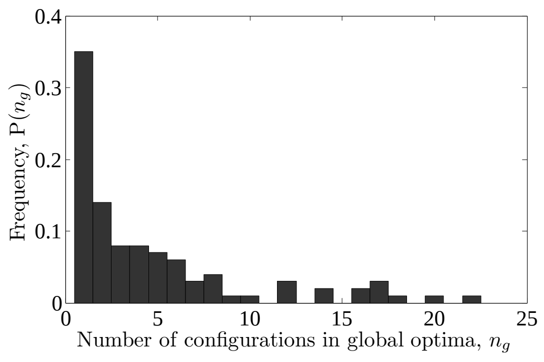
TABLE 3.4: Probability of finding global optima as a function of n for different values of k.

k	Probability of finding global optima
2	$1.17 \times e^{-0.038n}$
3	$2.17 \times e^{-0.060n}$
4	$3.21 \times e^{-0.072n}$
5	$5.59 \times e^{-0.084n}$
6	$12.18 \times e^{-0.097n}$
7	$22.01 \times e^{-0.106n}$
8	$27.11 \times e^{-0.104n}$

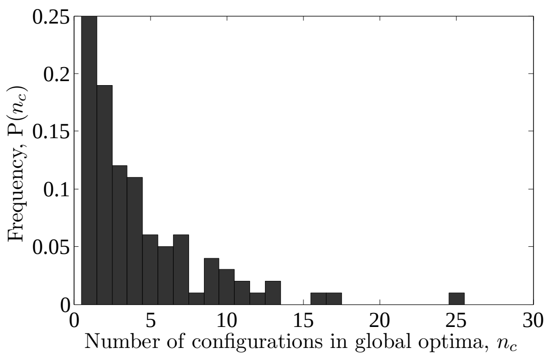
TABLE 3.5: Expected number of global optima for different n and k averaged over 100 problem instances.

k	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$
2	1.95	2.05	2.01	1.84	1.87
3	2.72	2.57	2.82	2.62	3.07
4	2.95	4.04	4.89	3.18	3.45
5	5.16	5.86	5.64	4.69	4.45
6	-	8.81	6.51	5.93	5.31
7	-	-	10.83	6.68	6.01
8	-	-	-	11.66	7.68

The expected number of global optima for different n and k is summarised in Table 3.5. It is clear that as k grows, the number of global optima increases, but it does not seem to have a particular relation with the size of the system. Here, again due to practical reasons, there is a limitation on the number of colours for which we can compute these quantities. Note that we define each optimum to be a connected set of configurations at the same cost with no fitter neighbours.



(a)



(b)

FIGURE 3.17: a) Histogram of the number of global optima for 10 000 random Graph-Colouring instances with $n = 50$ and $k = 5$. b) Histogram of the number of configurations at the global optima for 10 000 random Graph-Colouring problem instances with $n = 50$ and $k = 5$.

The size of the global optima also varies. Figure 3.17-b shows the histogram of the number of configurations at each global optimum for $n = 50$ and $k = 5$ for 10 000 randomly generated problem instances. The data are consistent with the number of local optima being distributed according to a geometric distribution,

$$\mathbb{P}(n_g) = (1-p)p^{n_g-1},$$

with $p \approx 0.75$.

The expected number of configurations in a global optimum is shown as a function of k and n in Table 3.6. This does not seem to change substantially with the number of vertices, n, but it increases with k. This is not surprising, as when we increase k, we move towards the phase-transition, where the sizes of all minima increase substantially. We are unable to measure the size of the global optima at or beyond the phase-transition as the number of states becomes too large.

TABLE 3.6: Expected number of configurations in global optima for different n and k averaged over 100 problem instances.

k	$n =$ 20	$n =$ 30	$n =$ 40	$n =$ 50	$n =$ 60
2	1.60	1.51	1.73	1.63	1.68
3	2.69	2.32	2.25	2.10	1.89
4	5.39	3.54	2.74	3.65	3.08
5	17.38	5.56	4.39	4.37	4.89
6	-	9.26	6.70	4.90	5.41
7	-	-	8.88	7.73	7.78
8	-	-	-	9.34	8.47

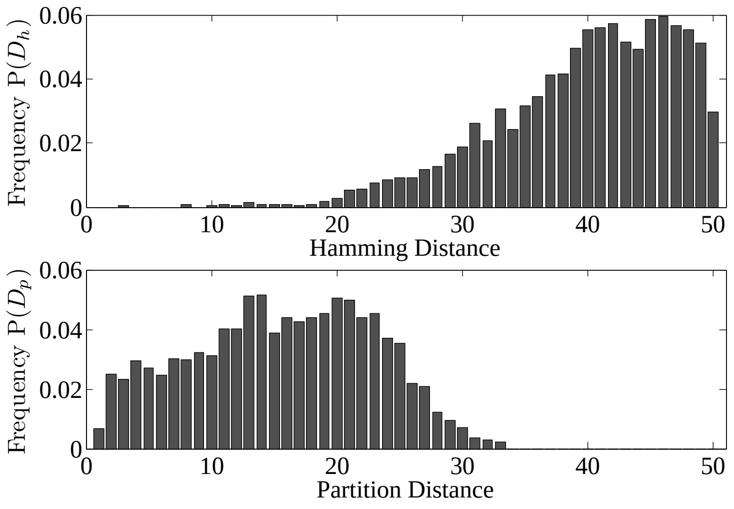


FIGURE 3.18: Histogram of the Hamming distances and Partition distances between the global optima for 10 000 random Graph-Colouring problem instances with $n = 50$ and $k = 5$.

Another important property of the fitness landscape of the Graph-Colouring problem is the distance between the global optima, which shows how the global optima are correlated. Figure 3.18 shows the histogram of the distances between global optima, averaged over 10 000 instances for $n = 50$ and $k = 5$, both in partition and Hamming space. For these parameters the expected Hamming distance between random configurations is 40, and the expected partition distance between random configuration is around 33. We see that the distances between global optima are widely distributed. Although there is a tendency for globally optimal solutions to be more correlated (in terms of partition distance) than random solutions, there exist cases where global solutions are very different from each other. Note that a fit solution is achieved through a choice of variables such that fewer than average constraints are violated. Because of the non-linear interactions between variables, this can be achieved in many different ways. The fact that the instances are drawn at random means that there are no strong statistical differences in the neighbourhood of the vertices. Thus it is not too surprising that there exist very different ways to obtain fit configurations. A consequence of the fact that there can be very fit configurations that lie a long way apart is that the position of the global optimum can be very sensitive to the exact distribution of edges. That is, by adding and removing an edge, the position of the global optimum can change to a completely different part of the search space. This makes it very challenging for a heuristic search algorithm to find a globally optimal configuration.

As we showed in Table 3.6, the number of configurations at the global optima increases as the number of colours grows. But one question that remains, however, is: what is the maximum distance between these configurations? Are they gathered together, or do they cover a wide Hamming region in the search space? Figure 3.19-a shows the mean Hamming and partition diameter of the global optima as a function of n for different number of colours. The Hamming or partition diameter of a local optimum is defined as the maximum Hamming or partition distance between every pair of configurations in the local optimum. It is clear in the Figure that the diameter of the global optima increases linearly with the system size. In Hamming

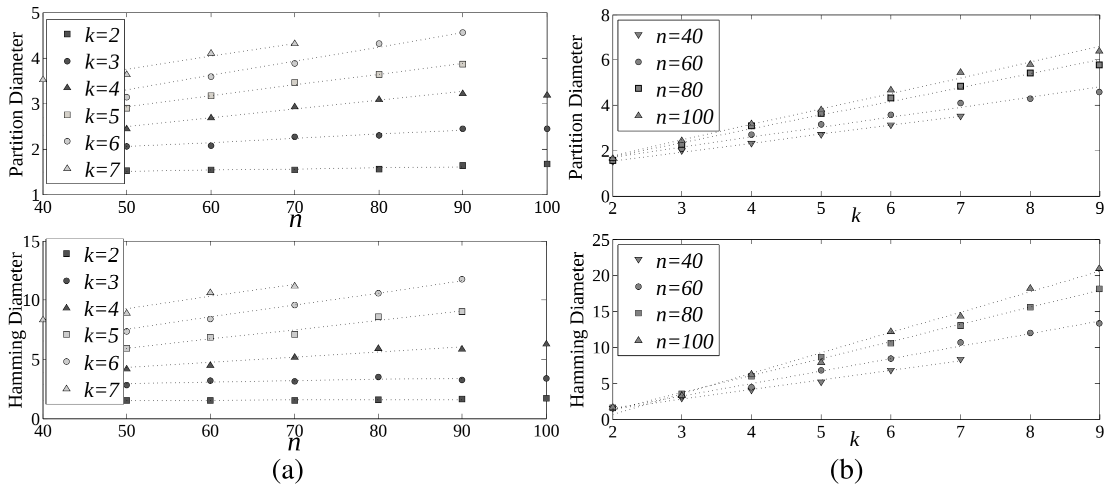


FIGURE 3.19: a) Average Hamming and partition diameter of global optima versus n for different number of k. The results are averaged over 1000 different random problem instances. b) Average Hamming and partition diameter of global optima versus k for different number of n. The results are averaged over 1000 different random problem instances.

space, for $k = 7$ and $n = 50$, the average diameter of global optima is 9 and for some problem instances a global optimum with a diameter of 20 is observed. It shows how wide the global optima could be.

There is also a linear relationship between the average diameter of the global optima with the number of colours. The Hamming and partition diameter of the global optima as a function of k are represented in Figure 3.19-b for different problem sizes. It is clear that the diameter of the global optima grows much faster with k than it grows with n. The Hamming and partition diameter of global optima as a function of k and n for different values of n and k are summarised in Table 3.7.

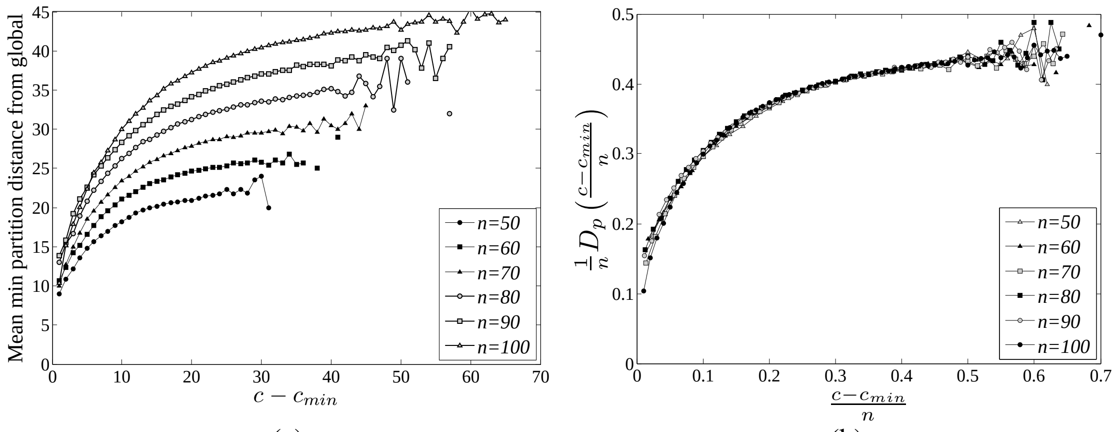
3.5.8 Distance Between Optima

In MAX-SAT problem we showed that there is a correlation between the cost of a local optimum and its distance to the closest global optimum. We show here that for the Graph-Colouring we see similar behaviour. Figure 3.20-a shows the mean minimum partition distance from a configuration in a local optimum to the nearest global optimum. These results are averaged over all the local optima found in 100 instances.

By rescaling the axes and fitting a single correction to scaling parameter, we can collapse the curves in Figure 3.20-a onto a universal curve. This is shown in Figure 3.20-b, and demonstrates that the distances from a local optimum to a global optimum scale linearly with the problem size. Making the last step from an optimum of cost $c_{min} + 1$ to c_{min} clearly grows with n, although it is difficult to be sure how this distance grows.

TABLE 3.7: Hamming and partition diameter of global optima as a function of system size n, and number of colours k, for different k and n respectively.

Number of Colours	Hamming Diameter	Partition Diameter
2	$1.39+0.002n$	$1.39+0.002n$
3	$2.41+0.011n$	$1.60+0.009n$
4	$2.14+0.043n$	$1.62+0.019n$
5	$1.97+0.079n$	$1.72+0.024n$
6	$2.46+0.102n$	$1.93+0.031n$
7	$4.02+0.104n$	$2.31+0.028n$
System Size		
30	$-1.48+1.44k$	$0.48+0.49k$
40	$-1.25+1.34k$	$0.73+0.39k$
50	$-1.68+1.52k$	$0.86+0.38k$
60	$-2.04+1.74k$	$0.83+0.44k$
70	$-3.24+2.14k$	$0.68+0.53k$
80	$-3.43+2.37k$	$0.52+0.61k$
90	$-3.66+2.48k$	$0.59+0.63k$
100	$-4.95+2.83k$	$0.40+0.69k$



(a) (b)

FIGURE 3.20: a) Measure of the mean partition distance to the closest global optimum from a local optimum of fitness c for $k = 2$ and different values of n. b) Rescaled version of (a), showing that the curves collapse on to each other.

3.6 Landscape Correlation

As we show in Section 6.1.1, there exist significant long-range correlations in the landscape of the Graph-Colouring problem. This is a consequence of the structure of the fitness function. On average each vertex is connected to $p(n-1)$ other vertices. The change in the cost of a configuration due to changing the colour of a vertex will be equal to the number of the neighbour vertices with the new assigned colour, minus the number of the neighbour vertices with the previous colour. Typically, this change in fitness is rather small. As a consequence, similar to what we saw in the MAX-SAT problem, the configuration around a fit configuration will also tend to be fit. This property is also reflected in Figure 3.11-b, where the density plot of the record of the steps to local optima is represented. Close to the optima, there are huge plateaux where any improving neighbour is likely to improve the cost by at most 1, meaning that close to a local optimum, there are typically a large number of configurations having a similar cost to the local optimum.

3.6.1 Expected Cost in Hamming Sphere

In Figure 3.21-a, the expected cost of a configuration in a Hamming sphere of radius h from a global optimum is shown. The quantitative shape of the curve is quite similar for different n, although the standard deviation is of order n^2.

The expected cost of a configuration in a Hamming sphere of radius h around a configuration h is equal to,

$$c_h(\boldsymbol{x}) = \frac{1}{|\mathscr{H}_h(\boldsymbol{x})|} \sum_{\boldsymbol{x}' \in \mathscr{H}_h(\boldsymbol{x})} \sum_{(i,j)\in\mathscr{E}} [\![x'_i = x'_j]\!] \, ,$$

where $\mathscr{H}_h(\boldsymbol{x})$ is the set of configuration at a Hamming distance h from $\boldsymbol{x}$. We can compute a good approximation to this by assuming that we independently mutate a vertex colour with a probability h/n. There are then three types of edges we have to consider.

1. Those edges where colour at the two vertices are unchanged (i.e., $x'_i = x_i$ and $x'_j = x_j$), which occur with a probability $(1-h/n)^2$. In this case, the cost of the edges is the same as the cost of the initial configuration. $c(\boldsymbol{x})$.
2. Those edges where the colour of one vertex is changed, which occurs with a probability $2(1-h/n)h/n$. In this case, there is a probability of $1/(k-1)$ of an edge that was not a colour conflict becoming a colour conflict.
3. Finally, there are those edges where the colour of both edges are changed, which occurs with a probability $(h/n)^2$. In the case, if the edge was in conflict, there is a probability of $1/(k-1)$ that the edge remains a conflict. If, on the other hand, the edge was not in conflict, there is a probability $(k-2)/(k-1)^2$ that the edge becomes a conflicting edge.

Taking these three types of edges into account we find,

$$\begin{aligned} c_h(\boldsymbol{x}) &= \left(1-\frac{h}{n}\right)^2 c(\boldsymbol{x}) + 2\frac{h}{n}\left(1-\frac{h}{n}\right)\Big(|\mathscr{E}| - c(\boldsymbol{x})\Big)\frac{1}{k-1} \\ &\quad + \left(\frac{h}{n}\right)^2 \left(\frac{c(\boldsymbol{x})}{k-1} + \Big(|\mathscr{E}| - c(\boldsymbol{x})\Big)\frac{k-2}{(k-1)^2}\right) \\ &= c(\boldsymbol{x}) + \frac{kh}{n(k-1)^2}\left(\frac{|\mathscr{E}|}{k} - c(\boldsymbol{x})\right)\left(\left(2-\frac{h}{n}\right)k - 2\right). \end{aligned}$$

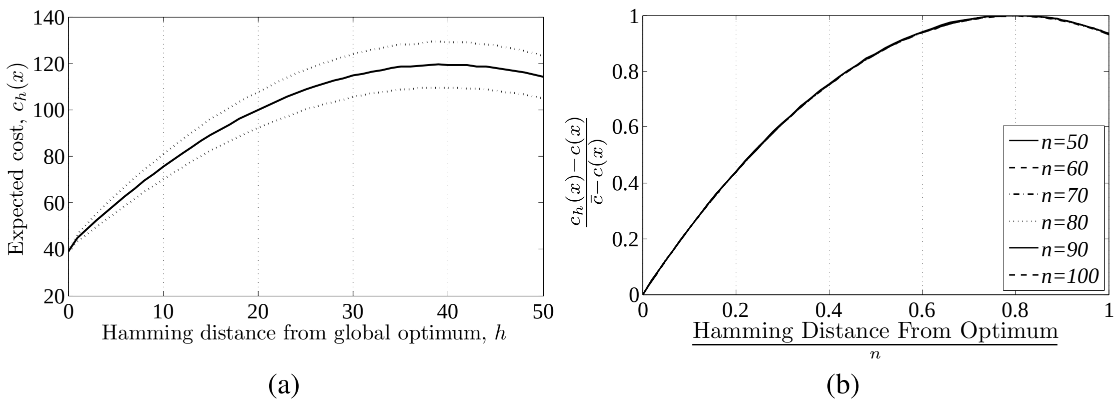


FIGURE 3.21: a) Expected cost of configurations in Hamming sphere of radius h around a global optimum for $n = 50$ and $k = 5$. The dotted lines show one standard deviation around the mean. b) Expected cost in a Hamming sphere for different sizes of the problem and for $k = 5$. Note that the minimum of the curves is at $h = n(1-1/k) = 0.8n$. The results are averaged over 10^4 solutions at Hamming distance h.

Since the average cost is equal to $|\mathscr{E}|/k$ we find,

$$\frac{c_h(\boldsymbol{x}) - c(\boldsymbol{x})}{\bar{c} - c(\boldsymbol{x})} = \frac{kh}{n(k-1)^2}\left(\left(2-\frac{h}{n}\right)k - 2\right).$$

This function reaches a maximum of 1 at $h = n(1-1/k)$. The rescaled cost for different n and for $k = 5$ is shown in Figure 3.21-b. This is computed empirically by sampling the cost 10^4 times at each Hamming distance, h. The theoretical approximation is indistinguishable from the empirical curves on this graph.

The same experiments are performed for different k and $n = 100$, and are shown in Figure 3.22-a. The curve for $k = 2$ has a complete symmetrical '$\cap$' shape. The reason for such behaviour is the symmetry in the search space; for $k = 2$, a fully mutated solution is identical to the original solution. So mutating all the variables in a configuration at a global optimum makes a solution at the same global optimum in the other symmetry.

Figure 3.22-b shows the expected cost for each Hamming sphere for the most frequently visited optimum at 9 different cost levels where an optimum was found. At each cost level, we chose the most frequently visited optimum. We note the same quantitative behaviour in all the curves, although there are some slight variations. As seen in this Figure, the expected cost in Hamming sphere is quite similar for all the local optima.

In Figure 3.23-a, a density plot of the local optima as a function of their fitness and their Hamming distance from the most frequently visited global optimum is shown. Apparently there is little correlation between the quality of the local optima and the closeness to this global optimum. Note that the local optima close to the global optimum are slightly below the average cost of local optima. The average Hamming distance for $n = 100$ and $k = 5$ is 80. It is clear that the peak of the density graph is at the left side of the line showing the average Hamming

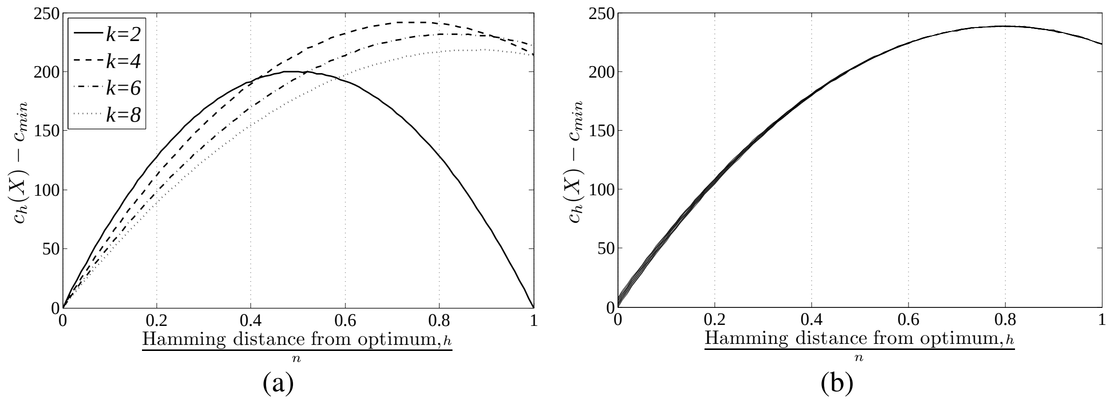


FIGURE 3.22: a) Expected cost in a Hamming sphere for different numbers of colours and for $n = 100$. Note that the minimum of the curves is at $h = n(1 - 1/k)$. The results are averaged over 10^4 solutions at Hamming distance h. b) Expected cost in a Hamming sphere for 9 different local optima for $n = 100$ and $k = 5$. The results are averaged over 10^4 solutions at Hamming distance h.

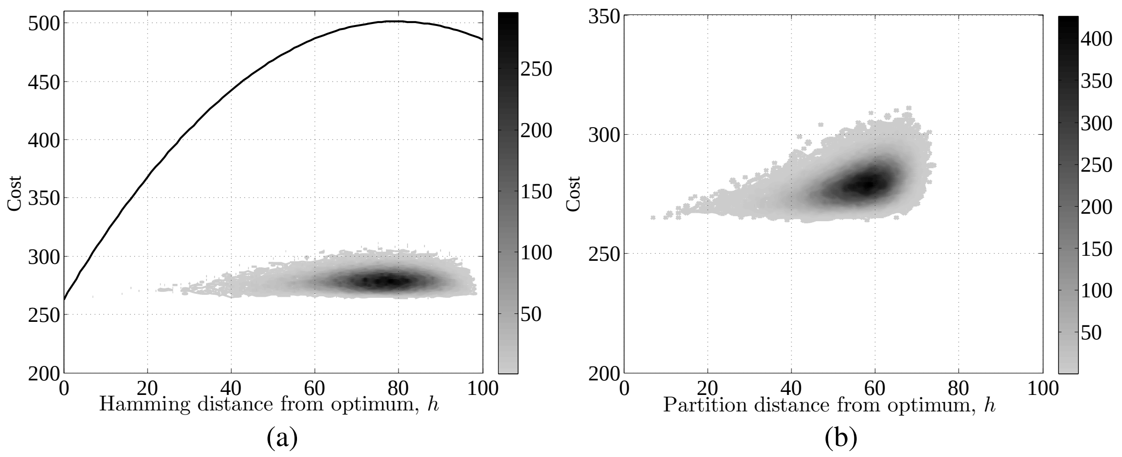


FIGURE 3.23: a) Density plot of local optima configurations as a function of their costs and Hamming distance from the most frequently visited global optimum. We also plot the mean cost in a Hamming sphere from the same global optimum. The shading of the point shows the number of configurations at that cost and Hamming distance. The size of the problem is $n = 100$ and the number of colours is $k = 5$. The number of Hill-climbs is 10^7 and the expected cost at Hamming distance h is found as an average over 10^5 configurations. b) Density plot of local optima configurations as a function of their costs and partition distance from the most frequently visited global optimum. The shading of the point shows the number of configurations at that cost and Hamming distance. The size of the problem is $n = 50$ and the number of colours is $k = 5$. The number of Hill-climbs is 10^7.

distance. For the case of Figure 3.23-a, 64% of the local optima have a smaller-than-average Hamming distance to the global optimum.

The same data are plotted against partition distance in Figure 3.23-b. The distances between the local optima in partition space show different behaviour from the Hamming space. The correlation between the quality of the local optima and the closeness to this global optimum is stronger using partition distance.

In order to show the relationship between n and k, with this correlation, we have computed the

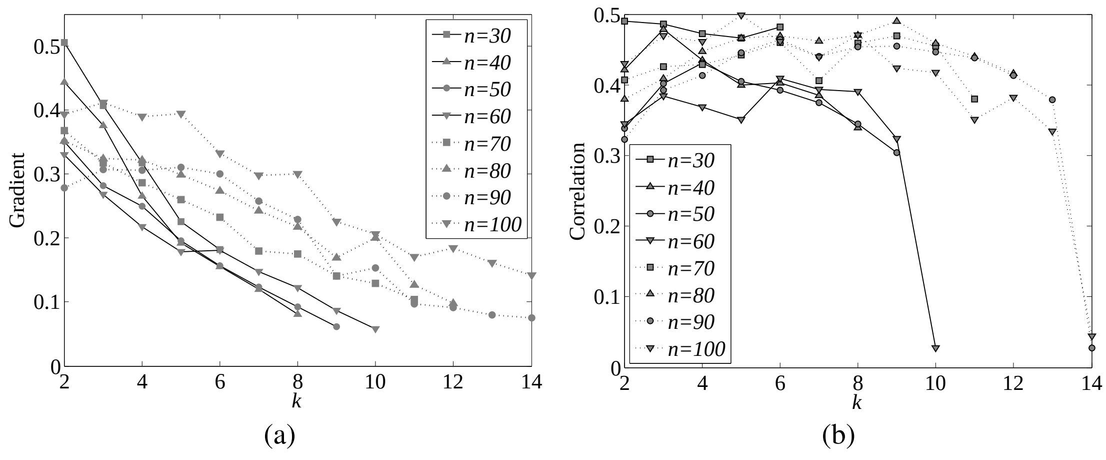


FIGURE 3.24: a) Gradient of the best fitting line to the cost of local optima as a function of their partition distance from the most frequently visited global optimum (see Figure 3.23-b). The data are averaged over 10^5 hill-climbs over 10 different problems at each problem size. b) Correlation between the cost of local optima as a function of their partition distance from the most frequently visited global optimum (see Figure 3.23-b). The data are averaged over 10^5 hill-climbs and over 10 different problems at each problem size.

gradient of the best fitting line to the density of the local optima as a function of their fitness and their partition distance. Figure 3.24-a shows these gradients for the most frequently visited global optimum. Each point represents the average over 10 random problem instances for 10^5 hill-climbs.

Another way to measure the relationship between the fitness of the local optimum and its partition distance to the most visited global optimum is to measure the (Pearson) correlation,

$$\mathrm{Corr}(c,D) = \frac{\sum_{i\in\mathscr{L}}\left(c_i - \bar{c}(\mathscr{L})\right)\left(D_i - \bar{D}(\mathscr{L})\right)}{\sqrt{\sum_{i\in\mathscr{L}}\left(c_i - \bar{c}(\mathscr{L})\right)^2 \sum_{i\in\mathscr{L}}\left(D_i - \bar{D}(\mathscr{L})\right)^2}},$$

where $\mathscr{L}$ is the set of local optima, c_i and D_i are the cost and distance to the global optima for local optimum, i, and $\bar{c}_{\mathscr{L}}$ and $\bar{D}_{\mathscr{L}}$ are the average cost and distances of the local optima. Note that this measure is closely related to the *fitness-distance correlation* (Jones, 1995b) (close because we consider the partition distance to the most frequently visited global optimum). The correlation for different values of n and k is represented in Figure 3.24-b. The data suggest a positive correlation in partition space, except when we reach the phase-transition and the global optima become very large. Below the phase-transition there appears to be no particular relationship between the correlation and the system size or the number of colours.

3.6.2 Exploiting Fitness-Distance Correlation

An important question for designers of heuristic search algorithms is if the correlation between fitness and distance of local optima can be exploited in some way. This is particular relevant for evolutionary algorithms where different members of the population may have found, or at least be close to, different local optima. One scenario where we might believe the fitness distance correlation can be exploited is in a hybrid generational genetic algorithm. In each generation we apply local search, selection and crossover. The local search will find good solutions (presumably near to local optima). As fitter solutions are closer (in partition distance) to global optima, this will move the population towards a global optimum. However, since there are an enormous number of local optima, the local search algorithms are unlikely to actually find the global optimum and will tend to get stuck. In selection we choose the fitter solutions, which should reduce the average distance to a global optimum because of the fitness-distance correlation. It does this at the expense of losing diversity. We then perform crossover, which helps restore diversity. Hopefully, the crossover will not (substantially) increase the distance to a global optimum, but it reinvigorates the search by moving the members of the population to a new part of search space away from a local optimum. Thus at the end of each generation, we should be closer to a global optimum than in the previous generation. As we lose diversity, due to selection, the search can "run out of steam" before it reaches a global optimum, although this can be mitigated to some extent, for example, by increasing the population size.

A critical part of the argument for the success of such a hybrid algorithm is that crossover does not substantially increase the distance to a global optimum. In most problems that are represented by strings, this is guaranteed by simple crossovers such as uniform crossover, or q-point crossover, provided that final population has essentially the same set of genetic variables after crossover (i.e., crossover just shuffles the variables between individuals). In this case, the average Hamming distance to any point in the search space is unchanged by crossover. However, in problems such as Graph-Colouring where there is an explicit symmetry, it is not sufficient that the average Hamming distance is unchanged, as there is no correlation between fitness and the Hamming distance. As a consequence, it has been long known that using a genetic algorithm with a traditional crossover operator is inefficient for Graph-Colouring. What we require is a crossover operator that minimises the change in the partition distance. Such an operator was proposed by Galinier and Hao (1999) over 10 years ago. Using a hybrid GA with this operator, Galinier and Hao obtained substantially better results on a large set of standard benchmark problems including large dense random problem. To the best of our knowledge, since then no new algorithm has been proposed with better performance on these standard benchmarks.

The description given above of how a hybrid genetic algorithm works appears plausible, but needs testing. We have attempted to do this by measuring the partition distance at each stage of a hybrid genetic algorithm using the Galinier and Hao crossover operator. In Figure 3.25 we show the minimum partition distance to the closest global minimum and the cost after local search, "L", selection, "S" and crossover "C" for an instance from $\mathscr{G}(100,1/2)$ with $k=3$. The results are averaged over 100 runs. Initially, the local search reduces the partition distance to a globally optimal configuration, but later on in the run, selection has a very significant influence in reducing this distance. Crossover significantly increases the average cost of the solution, but only increases the partition distance slightly. Local search rapidly repairs the damage caused by

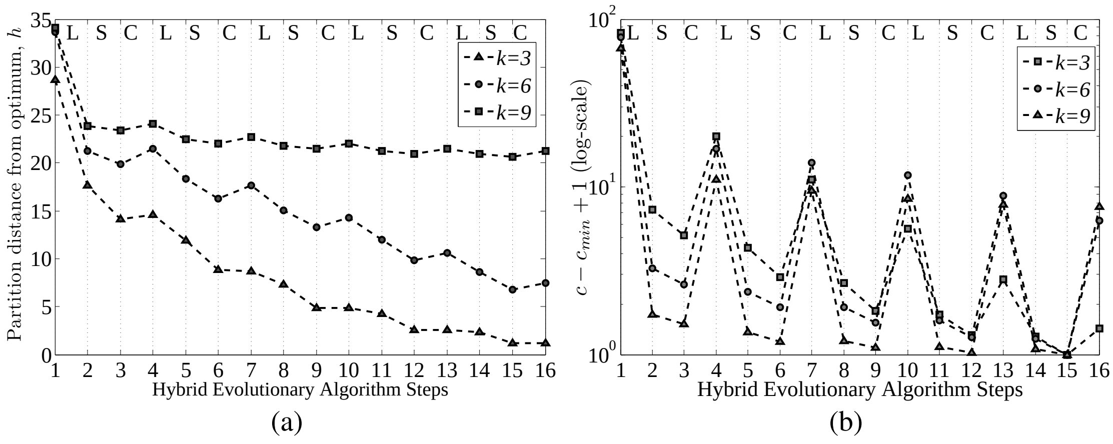


FIGURE 3.25: The evolution of a hybrid genetic algorithm where "L" represents the local-search algorithm, "S" represents the selection operator and "C" represents the crossover operator. This is collected on a randomly drawn instance of $\mathscr{G}(100, 1/2)$ with $k = 3$. The size of the population is 10. The results are averaged over 100 runs.

crossover. Thus the data shown support the picture presented above about how a hybrid genetic algorithm operates.

A landscape satisfies the big-valley hypothesis if the closer a local optimum is to the global optimum the fitter the optimum (Boese et al., 1994b). In the Graph-Colouring problem we often have many global optima that are far from each other. Furthermore, although there is a slight correlation between the fitness of a local optimum and its distance from a global optimum, there are plenty of fit local optima with a considerable distance from a global optimum, and many of unfit local optima close to a global optimum. Thus we cannot see the classic big-valley structure in the Graph-Colouring problem instances.

3.6.3 Basins of Attraction

The region in the search space that leads a local-search algorithm to a particular local optimum is usually called the "basin of attraction" of the local optimum. We perform the same method to study the basin of attraction of the local optima as we did for the MAX-SAT problem. Figure 3.26-a shows the return probability for the most frequently visited global optimum for a particular Graph-Colouring problem instance for different numbers of colours. As we move away from the minimum, the return probability initially decreases. Somewhat counter-intuitively it then increases again; however, this is because when we change enough variables we finish closer to an optimum that is identical up to a permutation of the colours—in all our work we have treated local optima that are identical up to a colour permutation as the same optimum. This is most pronounced in the case $k = 2$, where by changing all the colours we end up in the optimum obtained by swapping the two colours.

The return probability for the most frequently visited global optimum, and one of the least visited local optima is plotted in Figure 3.26-b. The most visited global optimum has been

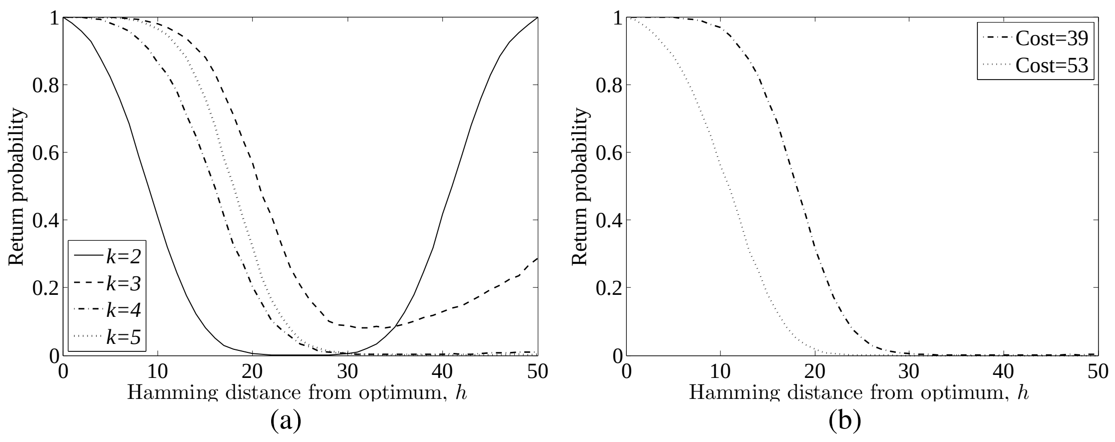


FIGURE 3.26: a) Return probability starting from a Hamming sphere of radius h versus h. This is for the most visited global optima in a particular graph with size of $n = 50$ for different number of colours. The empirical return probability is computed by running 10^5 hill-climbs starting at randomly chosen configurations in each Hamming sphere. b) Return probability starting from a Hamming sphere of radius h versus h. This is for the most visited global optima and the least visited local optima in a particular graph with size of $n = 50$ and $k = 5$. The empirical return probability is computed by running 10^6 hill-climbs starting at randomly chosen configurations in each Hamming sphere.

visited for 0.033% of the times, and the least visited one has been hit just once in 10^6 hill-climbs. We observe that the local-search algorithm has around a 83% chance of reaching the global maximum, if it starts within a Hamming distance of 20. This chance for the least visited local optimum is 51%.

3.7 Conclusion

One of the distinctive properties of Graph-Colouring is its permutation symmetry. This symmetry can be huge for large problems, for example, graphs from $\mathcal{G}(1000, 1/2)$ where the chromatic number is around 83, this permutation symmetry is truly vast ($83! \approx 4 \times 10^{124}$). The symmetry clearly has important consequences for any algorithm that tries to exploit long-range structure in the fitness landscape such as an evolutionary algorithm using crossover. Although low-cost solutions are moderately correlated with respect to the partition distance, they are not with respect to Hamming distance. Thus, we would expect that any effective crossover operator should minimise the partition distance between the children and the parents.

Another feature of Graph-Colouring problems that involves a large number of colours is the shear size of the search space. One consequence is that local optima tend to be some considerable distance apart. As a result local optima have a relatively large basin of attraction in terms of Hamming distances. The expected Hamming distance between random configurations is very large (precisely $n - n/k$), which means that even moderate correlation between configurations can be quite significant. This is seen when we perform Galinier and Hao crossover. As the parents tend to be quite far apart, the children are far from the parents and as a consequence can have a much higher cost than their parents. Nevertheless, the children tend to be in the basins of attraction of local optima with a similar cost to their parents. As a consequence, on running a local search algorithm, the child very rapidly reach the cost similar to their parents.

Chapter 4

Travelling Salesman Problem

In this chapter we study the landscape the Travelling Salesman Problem (TSP). The TSP is arguably the most famous combinatorial optimisation problem. The classic description of the problem is that you are given a list of cities and the distances between every pair of them. The task is to find the shortest tour that visits each city exactly once. Despite its deceptively simple description it is surprisingly difficult to solve. In general it is NP-hard, although for some problem types such as Euclidean TSP there is a polynomial time approximation scheme (Arora, 1998). In this research we investigate 11 different types of instances, many inspired by real world problems. The categories we have chosen have been studied in other papers (Rardin et al., 1993; Cirasella et al., 2001).

4.1 Problem Types

There are different ways of representing the TSP problem mathematically. One method is in terms of finding a bijection from each city to the successor city $s : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ which minimises the total tour distance

$$c(s) = \sum_{i=1}^{n} \mathbf{M}_{i\,s(i)} = \sum_{i,j=1}^{n} \mathbf{M}_{ij} [\![j = s(i)]\!]$$

where $\mathbf{M}_{ij}$ is the "distance" between cities i and j and $[\![\text{predicate}]\!]$ is an indicator function. For a legal tour we require $s(i) \neq i$. We consider 11 different problem types, which differ in the construction on the distance matrix $\mathbf{M}$. The only common assumption is that the distances, $\mathbf{M}_{ij}$, are not negative.

4.1.1 Random Asymmetric Matrices

In this problem the distance between the cities is chosen randomly,

$$\mathbf{M}_{ij} = \text{Rand}(0, 10^6), \quad \text{for} \quad i, j = 1...n, \tag{4.1}$$

where Rand(A, B) is a uniform random number generator that generates integer deviates, $\mathbf{M}_{ij}$, where $A \leq \mathbf{M}_{ij} < B$. Although this set of problems does not belong to the real world problems, many of the algorithm designer use this type as a challenge to their algorithms. We refer to this set of problems as the **Random** problems.

4.1.2 Random Asymmetric Matrices Closed Under Shortest Paths

The previous set of problems lacks the correlation between the distances, making this set an unrealistic problem type. Taking the random problems made through the previous method and closing them under shortest path computation make a more realistic set of problem. A matrix $\mathbf{M}$ is closed under shortest path if and only if,

$$\forall i, j, k, \;\; \mathbf{M}_{ij} \leq \mathbf{M}_{ik} + \mathbf{M}_{kj}. \tag{4.2}$$

That is, the "distances" satisfy the triangular inequality. Closing a random matrix can easily be performed by applying the following operator on all the values in the matrix,

$$\forall i, j, k, \; \textbf{if} \;\; \mathbf{M}_{ij} > \mathbf{M}_{ik} + \mathbf{M}_{kj} \;\; \textbf{then} \;\; \mathbf{M}_{ij} \leftarrow \mathbf{M}_{ik} + \mathbf{M}_{kj}. \tag{4.3}$$

This process is applied until all the values in the matrix satisfy the equation 4.2. We refer to this set of problems as the **Random C** problems.

4.1.3 Random Symmetric Matrices

This class of problems is like the Random problem, except the problem matrix is symmetric here,

$$\forall i, j, \quad \mathbf{M}_{ij} = \mathbf{M}_{ji}. \tag{4.4}$$

We refer to this set of problems as the **Random S** problems.

4.1.4 Random Symmetric Matrices Closed Under Shortest Paths

In this set of problems, the matrices are both Closed and Symmetric. We refer to this set of problems as Random SC problems. In this case, our distances between cities form a proper metric distance.

4.1.5 Random Two-Dimensional Rectilinear Problems

In this set of problems, the cities are uniformly distributed in a 10^6 by 10^6 square and the distance between the cities is computed based on the rectilinear metric,

$$\mathbf{M}_{ij} = |x_i - x_j| + |y_i - y_j|, \tag{4.5}$$

where (x_i, y_i) shows the coordinates of the i-th city. We refer to this set of problems as the **Rectilinear** problems.

4.1.6 Tilted Drilling Machine Instances with Additive Norm

This set of the problems are motivated by considering a the cost of driving an idealised drilling machine. The task is to drill a collection of holes on a tilted surface where the drill is moved by two motors. The first motor moves the drill in the x-coordinate and the second motor moves it in the y-coordinate. Because of the drill weight, the second motor needs less energy when moving the drill down, than moving it up. The problem generator places the holes uniformly in a 10^6 by 10^6 square and has three parameters, u_x is a coefficient which shows how much energy the first motor needs to move the drill one unit in the x direction, u_y^- is a coefficient which shows the energy needed to move the drill one unit down and u_y^+ which is the needed energy when moving upward. So the problem matrix is generated according to

$$\mathbf{M}_{ij} = \begin{cases} u_x|x_i - x_j| + u_y^+(y_j - y_i) & \text{if} \quad y_i \le y_j \\ u_x|x_i - x_j| + u_y^-(y_i - y_j) & \text{if} \quad y_i > y_j. \end{cases} \tag{4.6}$$

We consider u_x=1, u_y^+=2 and u_y^-=0. We refer to this set of problems as the **Additive Drilling** Problems.

4.1.7 Tilted Drilling Machine Instances with Sup Norm

For many drilling machines, the cost of drilling will depend on the maximum time the drill moves in either the x and y directions rather than their sum. For this problem type the holes are placed just like the previous problem but the distances are found as follows,

$$\mathbf{M}_{ij} = \begin{cases} \max\left(u_x|x_i - x_j|, u_y^+(y_j - y_i)\right) & \text{if} \quad y_i \le y_j \\ \max\left(u_x|x_i - x_j|, u_y^-(y_i - y_j)\right) & \text{if} \quad y_i > y_j. \end{cases} \tag{4.7}$$

For this problem type we consider u_x=2, u_y^+=4 and u_y^-=1. This problem type is referred to as the **Sup-Drilling** problem.

4.1.8 Random Euclidean Stacker Crane Problem

We consider a crane having to move a set of n objects. Each object is at a (source) location $\boldsymbol{s}_i$ and has to be moved to a destination location $\boldsymbol{d}_i$. The problem is to chose the order of the tasks to minimise the Euclidean distance between the destination location of one task and the source location of the next location. That is, we take the distance to be

$$\mathbf{M}_{ij} = \|\boldsymbol{d}_i + \boldsymbol{s}_j\|,$$

where $\|\cdot\|$ denotes the Euclidean distance. The sources are uniformly picked up from a 10^6 by 10^6 square,

$$\boldsymbol{s}_i = \left(\text{Rand}(0,10^6), \text{Rand}(0,10^6)\right), \text{ for } i = 1...n, \tag{4.8}$$

while the destinations are chosen to be "close" to the sources,

$$\boldsymbol{d}_i = \boldsymbol{s}_i + \left(\text{Rand}(0, \frac{10^6}{\sqrt{n}}), \text{Rand}(0, \frac{10^6}{\sqrt{n}})\right), \tag{4.9}$$

where n is the number of source-destination pairs (problem size). We denote this problem by **crane**.

4.1.9 Disk Drive Problem

These instances are generated by an idealised model of the scheduling job of the read head of a computer disk. The task is to extract n records from a disk. Each disk has a start and end position in their track. The position of the record is denoted by two coordinates. The first coordinate represents the distance along the track while the second position represents the track number. The source points are generated using equation 4.8 while the corresponding destination is generated using

$$\boldsymbol{d}_i = (s_{i1} + \text{Rand}(0,10^5) \mod 10^6, s_{i2}),$$

where $\boldsymbol{s}_i = (s_{i1}, s_{i2})$ is the coordinate of the destination. This models the situation where the record lies on the same track as the source and the disk has circular boundary conditions in the first coordinate. Furthermore, we assume that the head is moved in the second coordinate, but must wait until the spinning disk reaches the correct first coordinate. The head motion is assumed to be 10 times slower than the disk motion. The cost is taken to be the time to access all the records and return to the starting position. Although, this is clearly idealised (for example, no account is made of acceleration and deceleration times of the head), nevertheless, it captures much of the structure that a real disk problem would have. We refer to this problem as **Disk Drive**.

4.1.10 Euclidean Problem

This is the well-known Euclidean TSP. To generate instances we randomly select the cities in a $10^6 \times 10^6$ square, and the distance between the cities is the Euclidean distance between them. We refer to this problem type as **Euclidean**.

4.1.11 No-Wait Job-Scheduling

In this problem we assume that we have k processors and a set of n tasks, each of which must use the k processors in order with no waiting time allowed between ending one process and starting the next. The no waiting condition would be realistic, if, for example, the processing required the material to be kept hot, or alternatively, if there was no storage space between two consecutive processes. We denote the times taken to complete the l^{th} process for the i^{th} task as u_i^l. Thus each task can be represented by a list of sub-task times $\boldsymbol{u}^i = (u_1^i, u_2^i, \ldots, u_n^i)$. What makes the scheduling task non-trivial is that you can start the next task before the previous task is finished, provided that none of the sub-tasks have to wait. Thus the distance between tasks is the wait time

$$M_{ij} = \max_{l \in 1,2,\ldots,k} \left(\sum_{r=1}^{l} u_r^i - \sum_{s=1}^{l-1} u_s^j \right),$$

We illustrate this wait time for two tasks in Figure 4.1. In this research we consider $k = 5$ and the times u_i^l are made by a uniform random number generator between 1 and 1000.

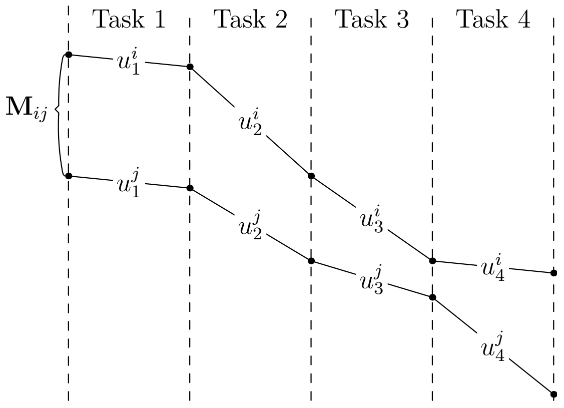


FIGURE 4.1: Example of the wait time between process $\boldsymbol{u}^i$ and $\boldsymbol{u}^j$ for a problem consisting of 4 processes. The waiting time, M_{ij} is caused by process j having to wait until process i has completed task 3.

4.2 Local Search Operator

To define the landscape of a problem requires some definition of the neighbourhood of the configurations. The neighbourhood is sometimes defined in terms of the configurations that can be reached by some local search operator. We consider k-opt moves which are the most commonly used move set by heuristic algorithms to solve TSP. A k-opt move consists of dividing the tour into k segments and then recombining the segments in such a way as to obtain a legal tour. The simplest k-opt move is 2-opt. For most of the analysis we carried out we found 2-opt impractical for two reasons. Firstly, for tours of moderate size (n around 50) we found that the number of local optima was so large that we were unable to store them. Secondly, the 2-opt move reverses one segment of the tour. For asymmetric problems this would disrupt a large number of edge distances so 2-opt cannot be viewed as a minimal move for this set of problems. Thus, in this research, we focus on 3-opt moves, which both substantially reduces the number of local optima and contain moves which change the tour, but does not necessarily change the direction in which a segment is traversed. We also considered using 4-opt, however, we found that this had such a large neighbourhood it became computationally very expensive to ensure that a local optimum had been reached.

Although, it is convenient to consider k-opt neighbourhoods, it is often more convenient to use the Hamming distance in the space of edges rather than the minimum number of k-opt moves needed to go from one configuration to another. Note that there is a difference in the interpretation of distance between symmetric and asymmetric problems. In symmetric problems the edges (i,j) and (j,i) are treated as the same, while they are treated as different edges for asymmetric problems. If we swap three bonds in a 3-opt, this would correspond to making a move of size 3 for symmetric problems. For asymmetric problems the distance would also depend on whether any of the segments of the tour were reversed or not.

4.3 Landscape Analysis

In this section we study some properties of the fitness landscape of the TSP and show the effect of the size of the problem on the properties of the landscape for different types of the problem. The results we describe in this research show why as the size of the problem grows, it becomes harder for local search algorithms to find optimal solutions.

4.3.1 Density of States

We start our analysis by studying the statistical properties of randomly drawn solutions in the landscape. In our experience these properties do not correlate well with the problem hardness, but they can be important in determining which search operations to use for a particular type of problem.

The density of states shows the number of configurations at each cost level. The average cost of the solutions in TSP is the average distance between the nodes. By randomly sampling the solutions and finding the histogram of cost of the random solutions, we can compute the spread of costs around the mean. Figure 4.2 shows the natural logarithm of the histogram of costs around the average cost scaled by $\sqrt{n}$ for $n =$ 100, 500 and 1000 for the Euclidean problem, where n is the size of the problem. The results are similar for every randomly drawn problem instance. The cost of the random solutions is approximately normally distributed around the average cost and the variance of approximately $\sqrt{n}$. We did the same experiment on all the 11 problem types. In each case we averaged over 50 different problem instances of each of type. We observed very similar behaviour for all the problem types.

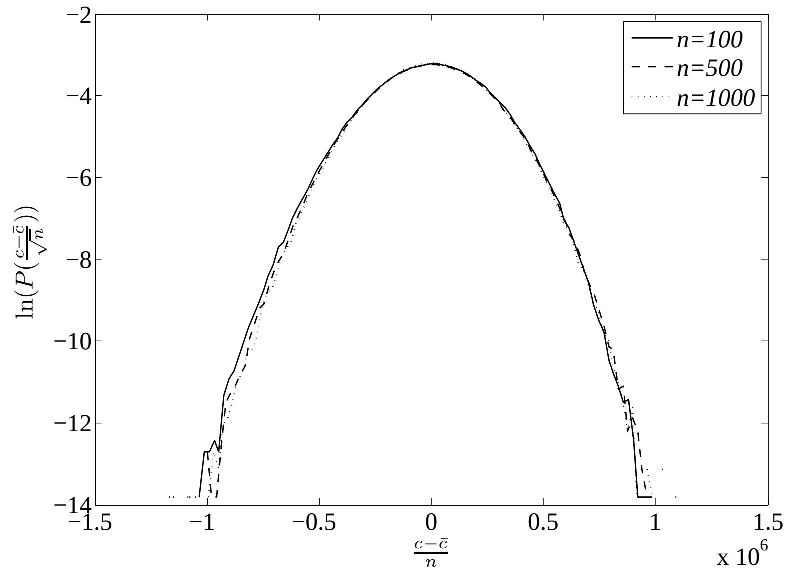


FIGURE 4.2: Natural Logarithm of histogram of costs for random solutions of particular instances of Euclidean problem for $n = 100, 1000, 10000$.

4.3.2 Statistical Measures

The density of states for all problems is a bell-shaped curve resembling a normal distribution. To quantify the difference between the problems, we can measure the cumulants, κ_n. The first cumulant is the mean, the second cumulant is the variance, while the higher cumulants provide a natural measure of the deviation from the mean. The third and fourth cumulant can also be expressed in terms of the central moments

$$\mu_n = \mathbb{E}\left((c - \mathbb{E}(c))^n\right),$$

where c is the cost of the tour and the expectation is over all instances of the problem. The third cumulant is equal to the third central moment μ_3 while the fourth cumulant is equal to the fourth central moment minus three, times the variance squared. In a normal distribution the third and forth cumulants are zero. The higher-order cumulants are invariant to changing the definition of the costs by an additive constant, but are not invariant to the scale of the cost. A more useful set of measures to understand how much a distribution differs from a normal distribution is to consider the scaled cumulants were we scale the n^{th} cumulant by a factor of $\kappa_2^{n/2}$. The scaled cumulants are now invariant to an affine transformation of the costs $c \rightarrow a\,c + b$ (for $a > 0$). The third scaled cumulant is known as skewness, denoted by γ_1, while the scaled fourth cumulant is known as the kurtosis and is denoted by γ_2.

In principal we could compute the cumulants *ab initio* (we did this for Graph-Colouring). For example, the mean is equal to

$$\kappa_1 = \mathbb{E}\left(\sum_{i,j=1}^{n} \mathbf{M}_{ij} [\![j = s(i)]\!] \right).$$

For random tours $\mathbb{E}([\![j = s(i)]\!]) = [\![j \neq i]\!] / (n-1)$ so that

$$\kappa_1 = \frac{1}{n-1} \sum_{i,j=1}^{n} \mathbf{M}_{ij} = n\,\bar{\mathbf{M}}$$

where $\bar{\mathbf{M}}$ is the mean of the distance values. We can thus compute the mean length of a random tour averaged over all instances by computing the average distance for the 11 problems. For example, for a 2-D Euclidean tour with cities randomly placed in a unit square

$$\begin{aligned}\bar{\mathbf{M}} &= \int_0^1 \int_0^1 \int_0^1 \int_0^1 \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}\, \mathrm{d}x_0\, \mathrm{d}x_1\, \mathrm{d}y_0\, \mathrm{d}y_1 \\ &= \frac{1}{15}\left(2 + \sqrt{2} + 5\sinh^{-1}(1)\right) \approx 0.521.\end{aligned}$$

Such calculations, however, are not very informative. Furthermore, for higher cumulants they rapidly become very complicated. We therefore do not pursue these calculations, but rather show empirically obtained results.

In TSP, as the size of the problem grows, the skewness and kurtosis of the cost of the random solutions change. Figure 4.3 shows the relationship between the skewness and kurtosis of the distribution of the costs for the Euclidean problem for different problem sizes. For small problems the skewness and kurtosis are negative. The negative skewness means that for small problems the bulk of the costs lies to the right of the mean and the negative kurtosis means that costs are more distributed around the mean than a normal distribution. The only problem type with a positive skewness is the Random problem. For Random Symmetric problems the skewness is around zero for all the problem sizes. As the size of the problem increases the skewness and kurtosis increase and become closer to zero, meaning that for larger systems, the distribution of solutions becomes closer to a normal distribution.

The skewness versus the kurtosis of the distribution of costs for different problem types for $n = 10$ and $n = 1000$ are represented in Figure 4.4. The results are averaged over 50 different problem instances and 10^6 sampling for each. It is clear that for all the problem types as the system size increases, the distribution of the costs converges towards a normal distribution. (This might seem an inevitable consequence of the central limit theorem, however it is not true, for example, in graph colouring in chapter 3.)

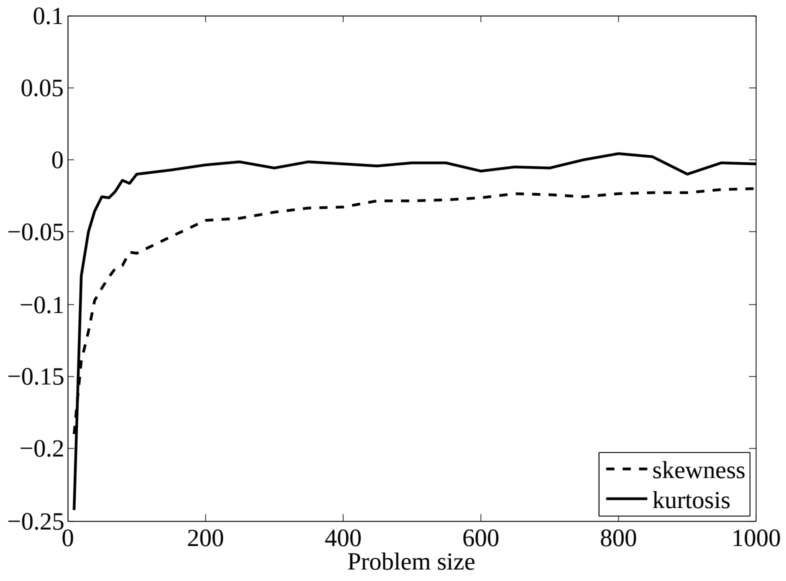


FIGURE 4.3: The skewness and kurtosis of the distribution of costs for Euclidean problem instances.

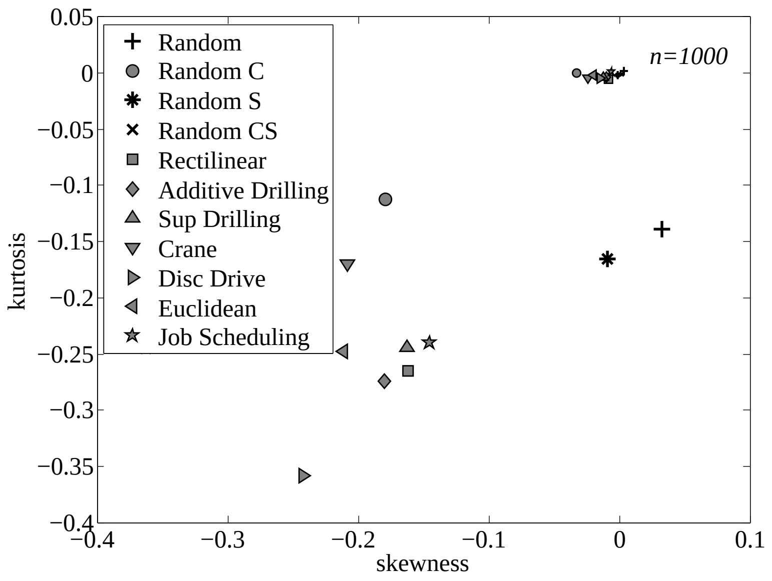


FIGURE 4.4: The kurtosis of randomly constructed solutions versus skewness for different problem types with different sizes. The small tags represent the problems with $n = 1000$ and the big tags represents the problems with $n = 10$. The results are computed by sampling 10^6 random configurations for 50 different problem instances.

4.3.3 Auto-Correlation

We have computed the auto-correlation for different sizes of different types of the problem. The curves representing the auto-correlation for particular instances of Euclidean problem for $n = 100$, 200 and 1000 for three different walks, 2-opt, 3-opt and 4-opt are shown in Figure 4.5. Note that for large values of τ the estimated value of $\mathscr{R}(\tau)$ is unreliable due to statistical fluctuations, even though we used 10^8 steps in our estimate. Under the rescaling used in this figure, the graphs for all the problems are very similar. The auto-correlation function appears to fall off approximately exponentially as,

$$\mathscr{R}(\tau) \approx e^{-\tau/l},$$

where l is known as the correlation length (Stadler, 1996). The gradient of the curves for 2-opt in Figure 4.5 is about -1.94, therefore for 2-opt $l \approx 0.51 \times n$. The correlation length shows that the landscape of TSP is relatively smooth with long range correlation. The landscape correlation will be discussed in section 4.3. It is also clear that for $k > k'$, the correlation between the solutions in k-opt is weaker than that in k'-opt. This is because as we increase k we are making larger steps and thus decorrelating faster.

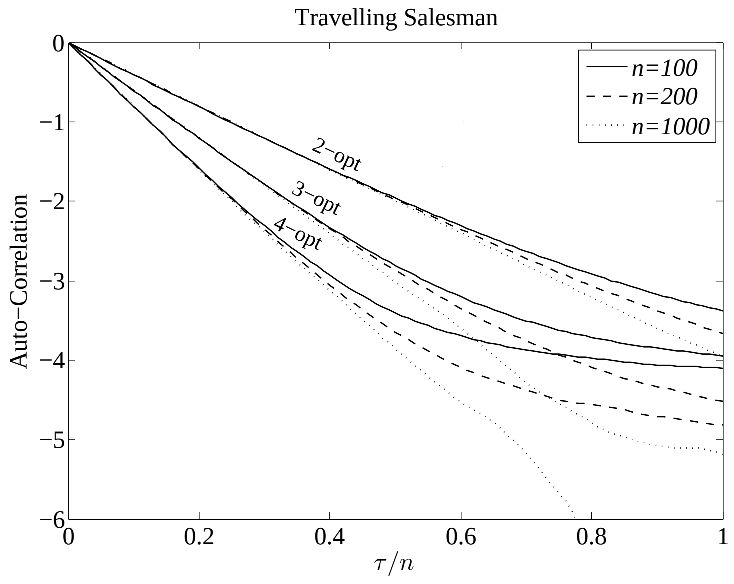


FIGURE 4.5: Natural logarithm of auto-correlation for three instances of size n =100, 200 and 1000, for Euclidean problem, plotted against the time difference τ/n. The number of steps is 10^8.

Figure 4.6 shows the auto-correlation function for all 11 problem types for $n = 1000$ where we use a 3-opt walk. We observe that the problems split into two major groups. For Random Symmetric, Random Symmetric Closed, Rectilinear, Additive-Drilling, Crane, Disc Drive and Euclidean problems the auto-correlation is a straight line, while for the Random, Random Closed, Sup-Drilling and Job Scheduling the curves have a distinct kink at the beginning. The kink appears at $\tau/n = 0.02$.

The kink occurs in those problems with asymmetric weights. The explanation of the kink is that typically a random 3-opt move will reverse the direction in which a tour is visited. For asymmetric problems this causes a rapid decrease in the correlation. However, later moves can reverse the direction that these cites are visited to the original direction, thus restoring the correlation.

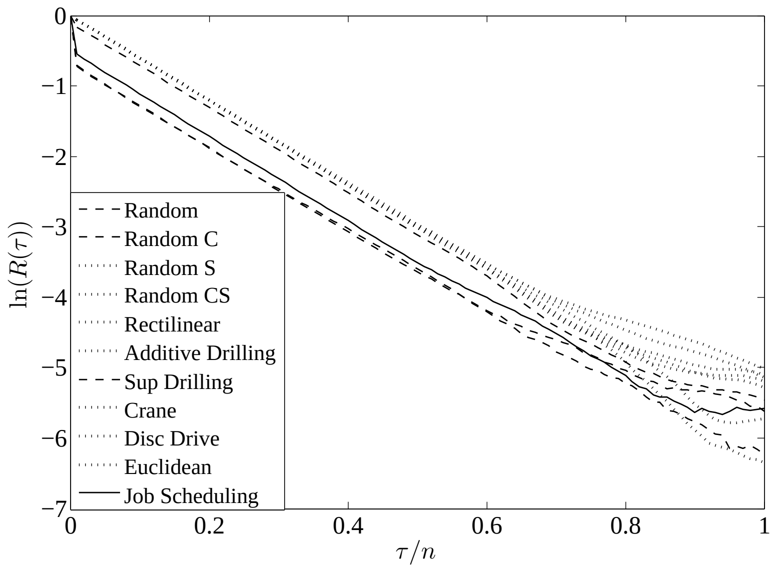


FIGURE 4.6: Natural logarithm of the auto-correlation for different problem types with the size of 1000, plotted against the time difference τ/n. The number of steps is 10^8. This is for 3-opt operator.

We can measure the degree of asymmetry of a problem using

$$A(G) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left|G_{ij} - G_{ji}\right|}{\sum_{i=1}^{n}\sum_{j=1}^{n} G_{ij}}. \tag{4.10}$$

In Figure 4.7 we show the correlation length for the first step versus the degree of asymmetry. We note that there are some problems, notably the Additive-Drilling problem and the Sup-Drilling problem, which have a reasonably high asymmetry, but do not show a marked decrease in the correlation produced by a single step. The reason for this is that they contain a hidden symmetry. If you reverse a segment of a tour for these problems the cost depends only on the difference in height of the starting and end point of the segment. As a consequence, these tours are much closer to the symmetric tours than the degree of asymmetry would indicate.

4.3.4 Time to Local Optimum

Our analysis begins by the study of the number of steps taken by the local search algorithm to reach a local optimum. We count the number of 3-opts performed on the starting configuration, until reaching a local optimum. Note that for symmetric problems, the 3-opt operator changes three edges at each step, and for asymmetric problems many of the edges may change. To check if we have reached a local optimum, an exhaustive search algorithm is used to make sure that the local search algorithm is at a local optimum.

We start our analysis by studying the relationship between the size of the problem and the time to reach a local optimum. Figure 4.8 shows the time to local optima versus n for all the problem types. It is clear that for all the problem types the time to local optima increases linearly with

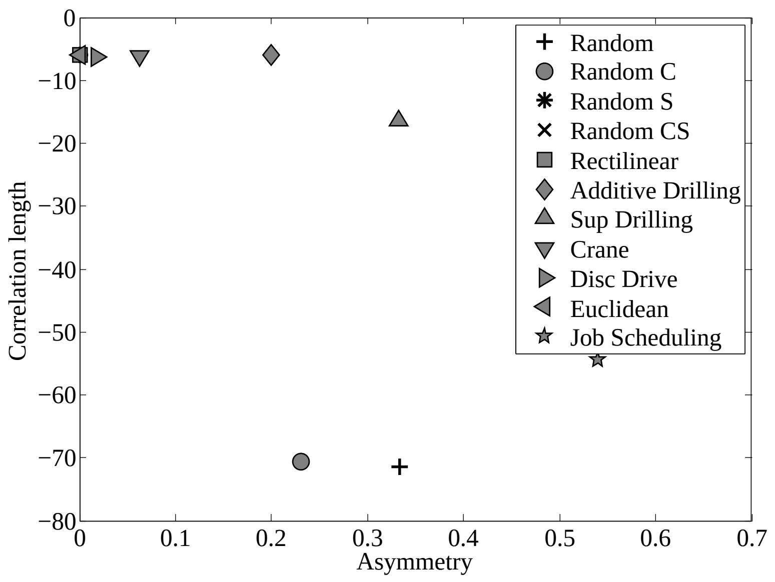


FIGURE 4.7: The correlation length for the first steps for different problem types against the asymmetry of the matrix. The size of the problem is $n = 1000$.

the system size (with the possible exception of Job Scheduling, which seems to grow slightly faster). The data suggest that the time to local optima increases as approximately $0.78n$. From the point of view of time to local optima, the problem becomes linearly harder as the system size grows. But of course it is not only the time that matters; the number of local optima is another important property which determines how hard the problem is. We will discuss this in the next sections.

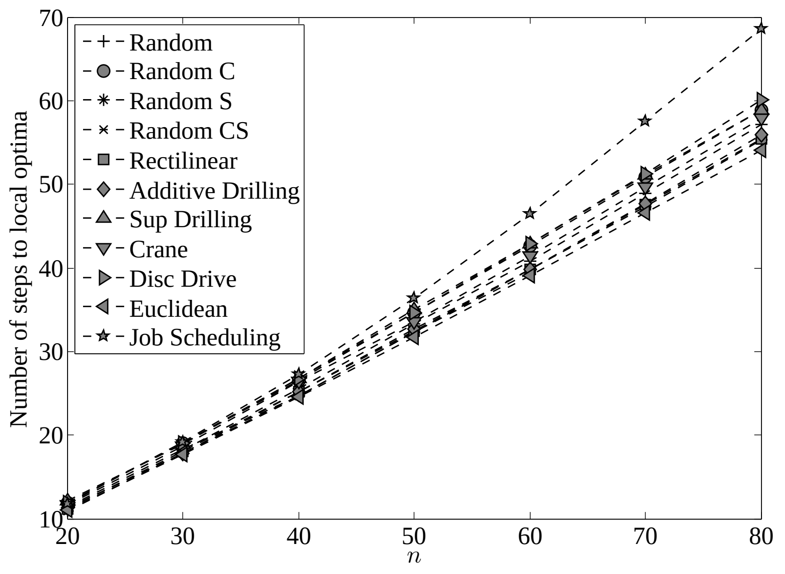


FIGURE 4.8: Time to local optima versus system size for different problem sizes.

Although the average time to reach a local optimum is quite similar for different problem types, the distribution of run times differs markedly between problem types. This is illustrated in Figure 4.9, where we show the histogram run times for the Sup-Drilling problem and the

Euclidean problem. We have plotted the log-frequency for the Sup-Drilling problem to show the rare events. Although the average number of steps is approximately the same for two different problem types (34.73 for Sup-Drilling and 31.67 for Euclidean), the Sup-Drilling problem shows a long tail at the right side of the mean, while the Euclidean problem shows a more symmetric shape with no long tail. The reason for such difference lies in the values of the distance matrices of the problems. For the Euclidean problem, the distances between the cities are real numbers, so even if truncated, it is exponentially rare to have two edges with the same length. This characteristic makes the fitness of almost all the solutions in the search space different from each other. Having such a characteristic, the landscape tends to have no plateau, so starting from a random solution; the local search algorithm directly reaches a local optimum. For the Sup-Drilling problem, the edges in the graphs take integer values, consequently there can be many different tours of the same length. This produces plateau regions which have to be explored, thus increasing the run time.

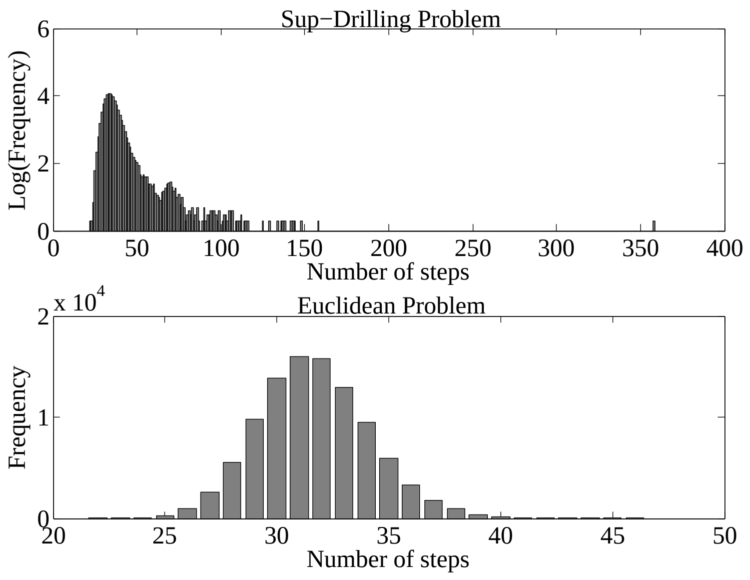


FIGURE 4.9: The histogram of number of steps to local optima. The size of the problem is $n = 50$. The data are for 10 different problem instances and 10^4 descents on each of them.

So far we have studied the number of steps to reach a local optimum, but we have not shown how the cost reduces during the run. In Figure 4.10 we show the cost versus the number of steps taken averaged over 1000 descents for an instance of the Sup-Drilling problem. We truncate the curves after the optima are reached. We plot the same data using a density plot and on a logarithmic scale in Figure 4.11. The most prominent characteristic is that the improvement in cost decreases roughly exponentially with the number of steps. However, there is clearly, a large deviation from this for a large number of steps caused by the relatively small number of runs that take a long time. These characteristics are shared by all problems, although the exceptionally large tail is a feature only of those problems with large plateaux.

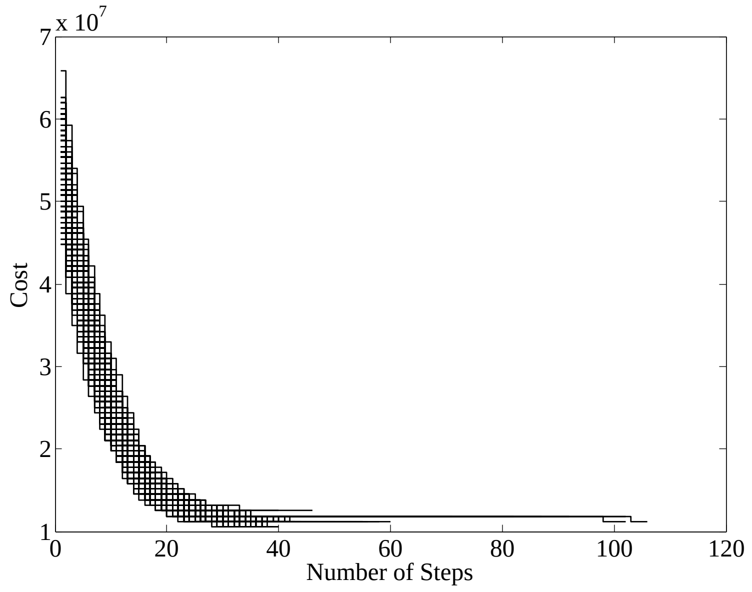


FIGURE 4.10: Cost of the solutions the local search algorithm visits at each step, for 1000 descents. This is for Sup-Drilling problem for the size of $n = 50$.

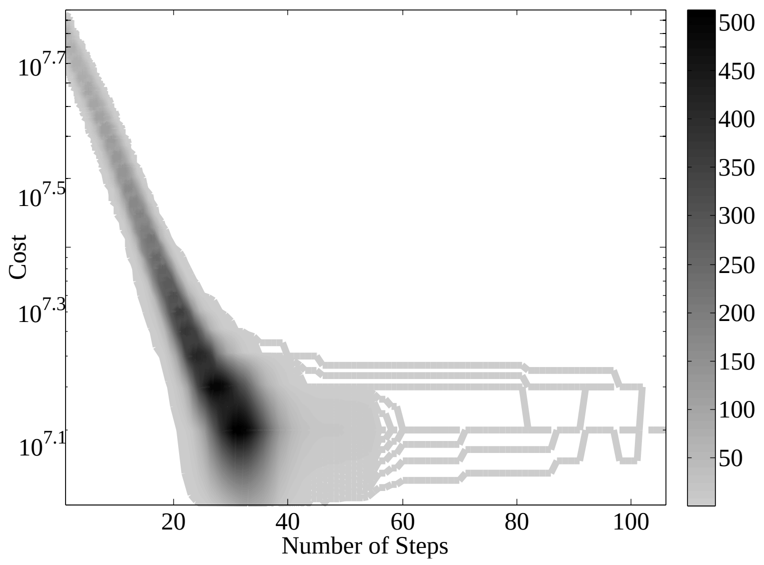


FIGURE 4.11: Density graph of logarithm of the cost of the solutions the local search algorithm visits at each step, for 1000 descents. This is for Sup-Drilling problem with the size of $n = 50$.

4.3.5 Number of Local Optima

What makes many combinatorial optimisation problem instances difficult is the number of local optima. To find the number of local optima in the landscape we use the exhaustive local search algorithm and store the local optimum it returns. The exhaustive local search algorithm is repeated for a large number of times from randomly chosen starting configurations. Each local optimum with the number of times it is hit is stored. Of course, there is no guarantee that all the local optima in the landscape are found, particularly if a local optimum has a small basin of attraction. Typically there are many local optima which have a very small probability of being visited.

The logarithm of the number of times the local optima are hit versus the cost of the local optima for the Euclidean problem is represented in Figure 4.12. The data are made based on one single instance and 10^5 descents. Of course there are likely to be local optima that we have not found. However, note that local optimum with the highest cost (the worst local optimum) is visited 18 times. The minimum number of times a local optimum is visited is 13. Thus with high probability any local optimum with a cost more than the worst local optimum that we have not found would have a basin of attraction 18 times smaller than those that we have found. Although we cannot rule out the existence of such local optima theoretically or practically, we strongly doubt their existence as we have not found any example of them in a very large number of trials.

We observe that the local optima that we have found in Figure 4.12 have a reasonable straight line fit of -7.29×10^{-6} which implies that the probability of ending up in a local optima of cost c decays exponentially as $P_v(c) \propto 10^{-0.00000729c}$. Using the extrapolation we find for the 10^5 descents, the expected number of times we would visit a local optima with a cost greater than 6.6×10^6 would be less than 1, if any such optima exist. The probability of finding the global optimum was 0.167 while the probability of finding the second best optimum was 0.149.

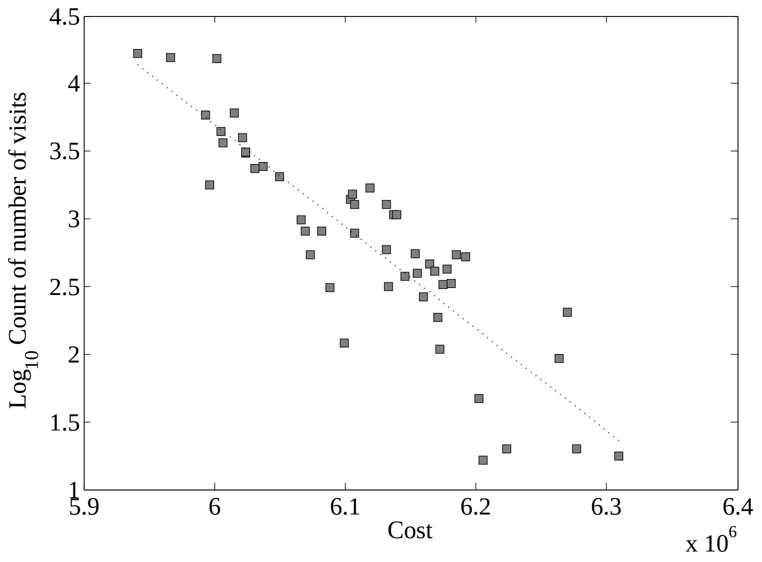


FIGURE 4.12: Logarithm of the number times the local search algorithm hits local optima versus the cost of the local optima. The size of the problem is n =50. This is for one instance of the Euclidean problem. The number of descents is 10^5.

Although for all the problem types the probability of getting to a local optimum decays exponentially as the cost of the local optimum grows, the decay rate is different for different problem types. Since the average cost of the local optima is different for different problem types, in order to be able to compare the decay rate of different problems, the cost of the local optima has to be normalised. To do so, the cost in the horizontal axis in Figure 4.12 is divided by the average cost of the local optima. After normalization, the gradient of the fitting line to the data shows the decay rate of the probability of getting to a local optimum as a function of its cost. The probability of getting to the global optimum versus the decay rate for all the problem types is shown in Figure 4.13. For the Random problems we found the best solution on average 1.07 times in 10^5 descents. Thus for this problem, we can have no confidence that we have found the global optimum. There seems little evidence of a relationship between the cost of the optima and the size of their basin of attraction. When the Random problems become symmetric, the probability of getting to the global optimum grows; on average the best solution is hit 20 times. When the Random problems are forced to satisfy the triangle inequality (Random Closed) the correlation between the cost of a local optimum and the probability of it being visited increase further. For this problem type, the average number of times the best found optimum is hit is 80 times in 10^5 descents. Going from Random problems to Random Closed problems, the decay rate grows as well. When the Random problems are both symmetric and closed, then the number of times the global optimum is hit grows to 4 119 times in 10^5 descents. For the case of Job-Scheduling problem, the best observed solution on average is hit 4 times in 10^5 descents. It is interesting, as the Job-Scheduling problem is not a random problem, we might expect observing a correlation between the cost of a local optimum and the size of its basin of attraction, but Figure 4.13 shows no such relation.

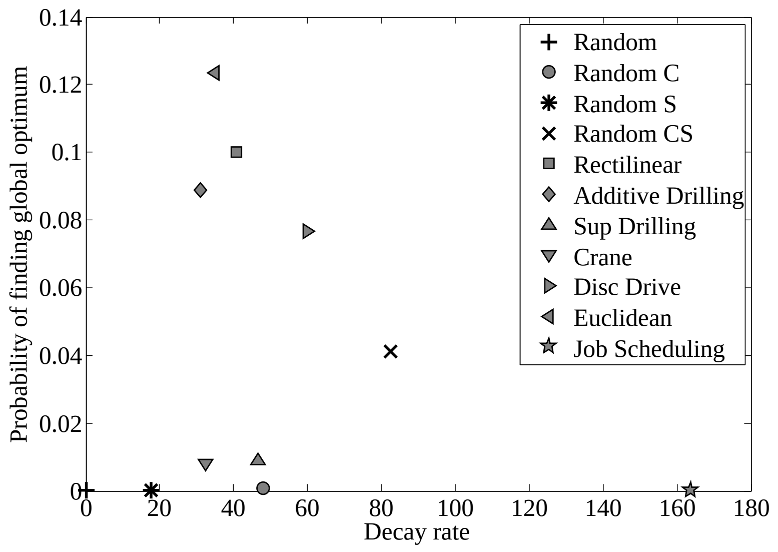


FIGURE 4.13: The probability of getting to the global optimum solution versus the decay rate for all the problem types. These are averaged over 10 different instances and 10^5 descents on each.

Problem types Sup-Drilling and Crane show a quite similar behaviour. For the Sup-Drilling problem the global optimum on average is hit 695 times and in the Crane problem it is hit 853 times in 10^5 descents. For the four remaining types the decay rate is similar but the probability of getting to the global optimum is different. The Euclidean problem shows the highest probability of getting to the global optimum. In the case of Figure 4.12 the best observed optimum is hit 16 752 times in 10^5 descents.

The histograms of the number of local optima at each cost for two different problem types are shown in Figure 4.14. The histogram shows different behaviour for different problem types; for the Euclidean problem it is close to a uniform distribution, while for the Sup-Drilling problem it is close to a normal distribution. In some cases like Job-Scheduling, the distribution has a long tail at the right side. A tail at the right side of the distribution means that there could be high cost local optima but with smaller probability of being visited. The distribution for the Euclidean problem shows no tail at either of its sides. There is quite a large proportion of local optima at both high and low cost levels.

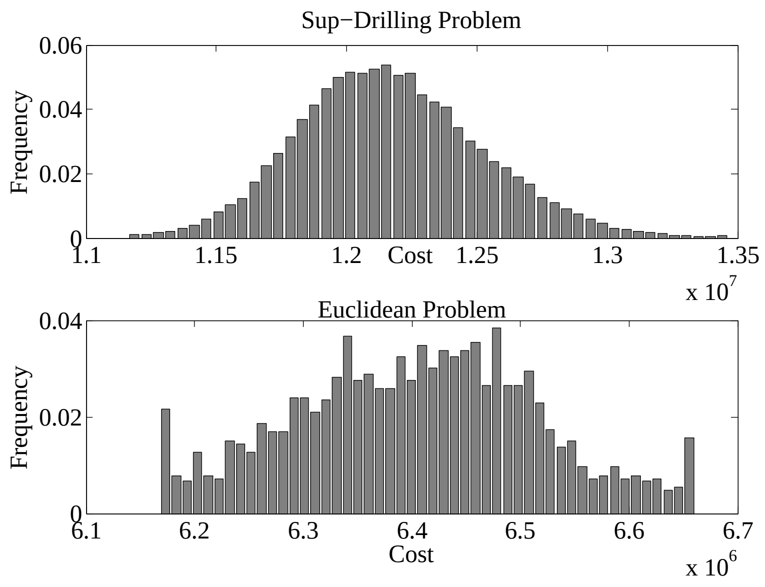


FIGURE 4.14: The histogram of the number of local optima at each cost, for two different problem types. The size of the problem is n =50. The data are for 10 different problem instances and 10^4 descents on each of them.

One way of measuring the size of the tails is to consider the distribution of costs of the local optima and measure the proportion of local optima in the 10% of highest and lowest costs. This is shown in Figure 4.15. There is clearly a strong correlation indicating some degree of symmetry. The Euclidean problem has the highest proportion of local optima at the extremes of the cost distribution, while the random problems have the lowest proportion.

4.3.6 Growth in the Number of Local Optima

The number of local optima grows with system size. If we take the logarithm of the number of local optima as a function of the cost and scale by a function of the system size we obtain very similar profiles. This is illustrated in Figure 4.16 where we show the natural logarithm of the number of local optima at each cost, divided by $n\ln(n)$ for the Euclidean problem for different problem sizes. The results are averaged over 20 different problem instances and 10^4 descents on each. This suggests that the number of local optima at each cost level grows exponentially as $N(c) \approx e^{n\ln(n)g(\frac{c-c_{min}}{c_{max}-c_{min}})}$, where $g(x)$ is roughly quadratic as shown in Figure 4.16. Figure 4.12, shows as the cost of a local optimum increases, the probability of the local search algorithm

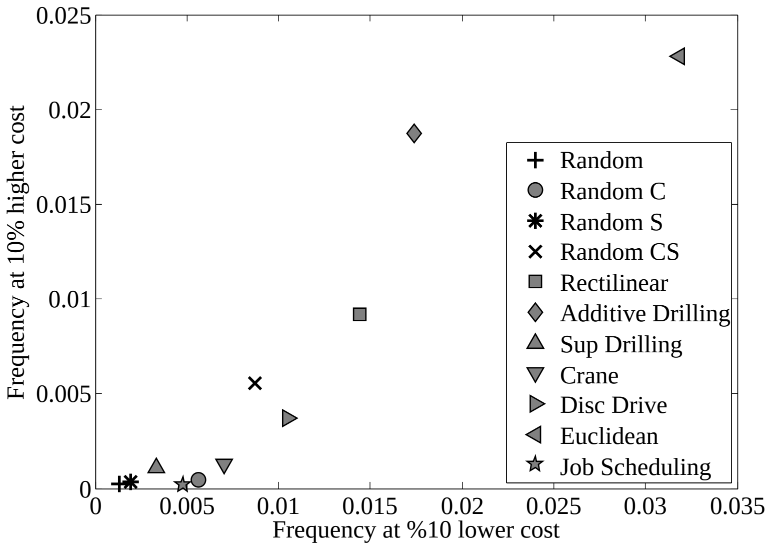


FIGURE 4.15: The frequency of the local optima at the higher 10% costs versus the lower 10% costs. The size of the problem in n =50 and the results are computed by sampling 10^4 descents for 50 different problem instances.

finding it decreases exponentially, so the number of local optima at higher costs (right part of the graph) is underestimated both in Figure 4.14 and Figure 4.16. The plot shows that the histograms are quite similar for different system sizes. Such behaviour is consistent with the hypothesis that the number of local optima grows exponentially with $n\ln(n)$. In order to fit the histogram for different system sizes, in the case of Euclidean problems, the natural logarithm of the number of local optima has been scaled by $n\ln(n)$ (note that the size of the search space also grows as $n! = \exp(\Theta(n\log(n))$. Although the number of local optima for all the problem types grows exponentially by system size, for other problem types the scaling behaviour changes. For example for Rectilinear, Disc-Drive and Job-Scheduling problems the scaling is n and for the Sup-Drilling and Crane problems is $\sqrt{n}$. For the Random problems (type 1 to 4) a vast majority of the local optima are hit once in 10^5 descents, so the number of local optima we find is almost equal to the number of descents, and it is hard to find the relationship between the system size and the number of local optima. It means that although for all the problem types the number of local optima increases exponentially with the system size, the growth rate changes from type to type.

It is interesting to note that the difficulty of the problem is not obviously correlated with the growth in the number of local optima. For example, Euclidean TSP has a larger probability of finding the global optimum than Sup-Drilling despite have significantly greater growth in the number of local optima than Sup-Drilling.

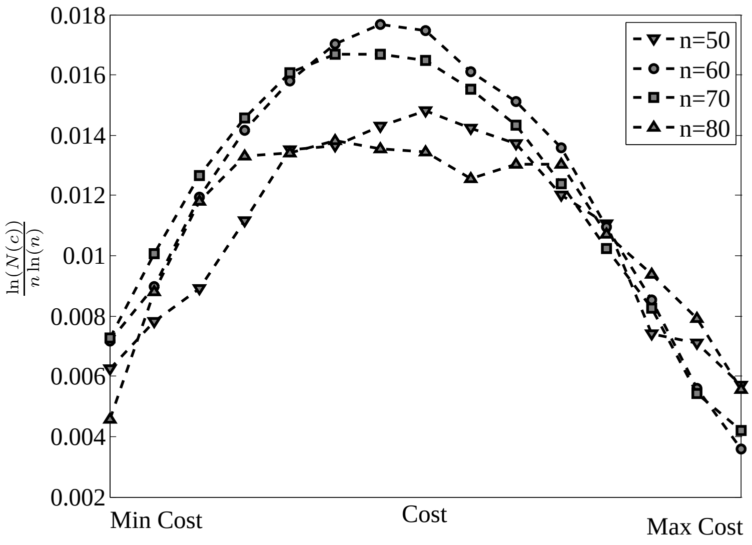


FIGURE 4.16: Natural logarithm of the number of local optima at each cost divided by n for Euclidean problem for different problem sizes. This is averaged over 20 problem instances and 10^5 descents on each.

4.3.7 Probability of visiting an optimum

We have seen that for some problems the probability of finding a local optimum is correlated with its cost, while this is not true for other problems. We illustrate this in Figure 4.17 which shows the proportion of local optima at each cost level and the probability of finding local optima at the cost level. For the Sup-Drilling problem, the lower cost solutions tend to have significantly larger basins of attraction. In contrast, in the Job-Scheduling problem the sizes of the basins of attraction are almost uncorrelated with the cost.

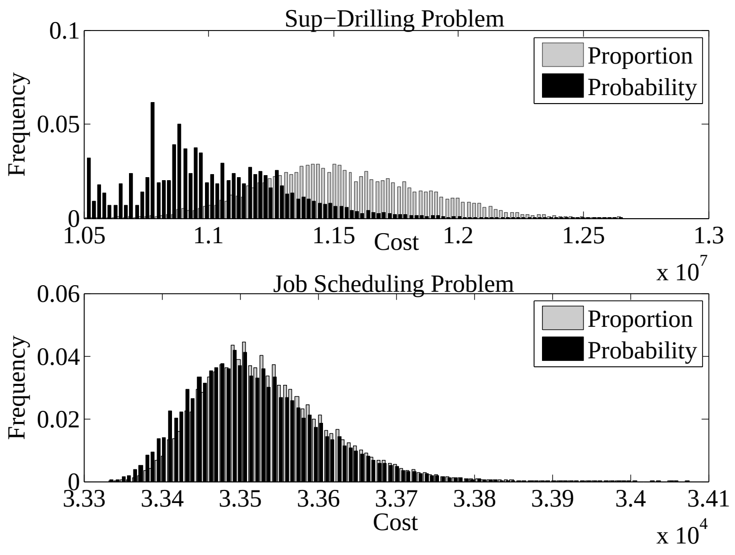


FIGURE 4.17: Histograms showing the proportion of local optima at a particular cost and the probability of finding a local optimum at a particular cost one instance.

Clearly, it makes the problems easier to solve when their basins of attraction are larger for lower cost solutions. To measure how the size of the local optima changes with the cost we define the *bias* to be the difference between the expected cost of all the optima and the expected cost of the optima found by local search, divided by the standard deviation in the cost of all the optima. A large bias, indicates that the local search is much more likely to find a fit local optimum than a less fit one. Figure 4.18 shows the probability of getting to global optimum versus the bias averaged over 10 different instances for each problem type. There is clearly some correlation between the probability of finding the global optimum and the bias, but interestingly this is not that strong.

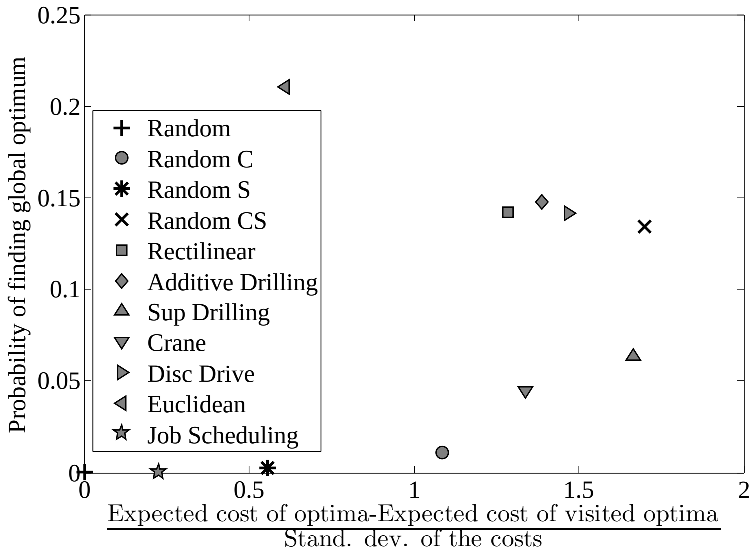


FIGURE 4.18: The probability of getting to the global optimum versus average of the cost of the local optima minus the expected cost of the visited local optima divided by the standard deviation of the cost of the local optima for all the problem types. The size of the problem is n =50. The data are averaged over 10 different problem instances and 10^5 descents on each.

4.3.8 Reaching the Global Optima

As we increase the system size the gap between the expected cost found by the local search and the minimum cost increases. This is illustrated in Figure 4.19, where the expected minimum cost (averaged cost of the global optima over 20 problem instances) and the expected cost of the local optima found by the local search algorithm over the same ensemble of randomly drawn instances are shown. We also show the expected cost of the local optima found by local search plus and minus one standard deviation. It is clear that as the system size grows the gap between the expected cost and the cost of the global optimum solutions grows. For problem sizes n up to 30, the cost of the global optimum solution lies between the expected cost of the local optima minus one standard deviation. For the size of 80, it lies between the expected cost and expected cost minus two standard deviation. Clearly as the system size grows it becomes less likely to find the global optimum.

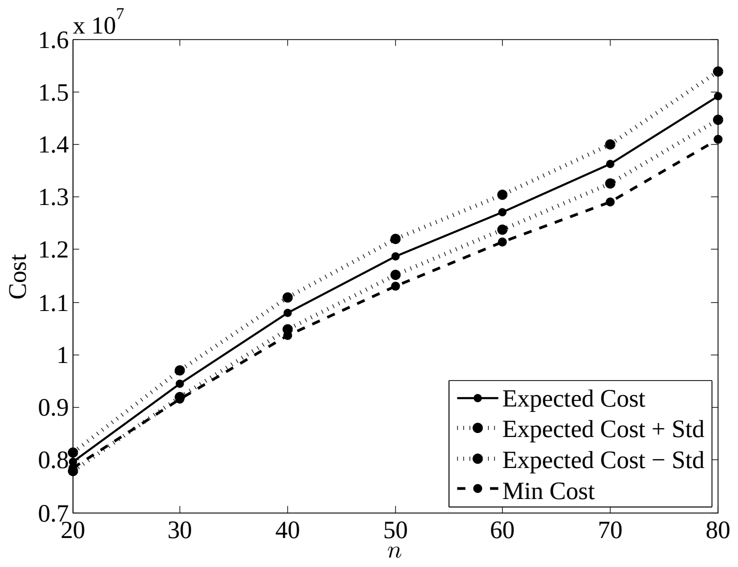


FIGURE 4.19: Plot of the expected minimum cost and the average cost of the local optima found by 3-opt versus n. We have computed the standard deviation in the cost and show the expected cost plus and minus 1 standard deviation. This is found by performing 10^5 hill-climbs on 20 randomly generated instances of the problem for Sup-Drilling problem.

We can also compute the probability of finding the best found local optimum directly. This is shown in Figure 4.20 where we plot the log-probability of finding the best optimum versus system size for four different problem types. The data are averaged over 20 different problem instances and the number of descents is 10^4. The straight line fit is consistent with the hypothesis that finding global optima becomes exponentially unlikely as the system size grows. Using the straight line fit in Figure 4.19 to extrapolate to large n, the probability of the local search algorithm finding a global optimum for an instance of size 1000 of the Euclidean problem would be 6.43×10^{-23} and 1.62×10^{-38} for the Sup-Drilling problem. Although such extrapolation is unlikely to provide a precise value, nevertheless, it provides a strong indication that for larger problem instances, using multiple runs of local search algorithms begins to be useless. For random and job scheduling problems the number of times the best visited optimum is visited, rapidly shrinks to one in 10^4 descents. For these problem types, for n larger than 40, the best visited solution is found just once, so we can almost be sure that for this system size, there are better optima that have not been found in this number of descents. The data show that for all the problem types as the system size grows the probability of finding the global optimum solution decreases exponentially, although with different rates.

Figure 4.19 shows that the gap between the expected cost and minimum cost increases with the system size and Figure 4.20 shows the probability of finding the global optimum decreases exponentially with the system size. These two figures show two different aspects of the same property of the fitness landscape. This property shows how the problem becomes harder as the system size grows. In order to show this for all the problem types, Figure 4.21 shows the gradient of the fitting line in Figure 4.20 versus the increasing rate of the gap in Figure 4.19 with the system size. The data clearly show the correlation between the gap and the probability of finding

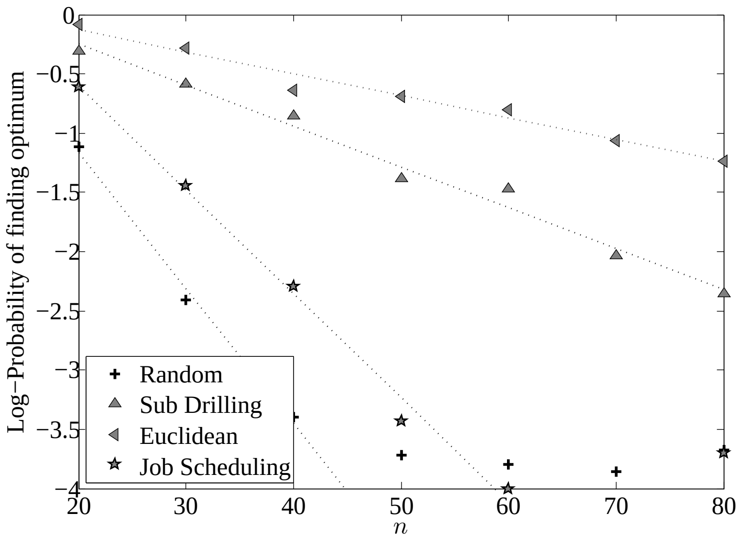


FIGURE 4.20: Log-Probability of finding the lowest cost local optimum for four different problem types versus system size n. The number of hill-climbs is 10^4 and the data are averaged over 20 different random problem instances.

the global optimum. The Job-Scheduling problem and the Random problem can be considered as the hardest problems, as the decay rate of the probability of finding global optimum is the highest among all the problem types. When the Random problems become symmetric or closed, the decay rate decreases, meaning the Random Symmetric and the Random Closed problems are easier than completely Random problems. Among these problem types, the Additive-Drilling and the Euclidean problem which show similar behaviour are the two easiest problems.

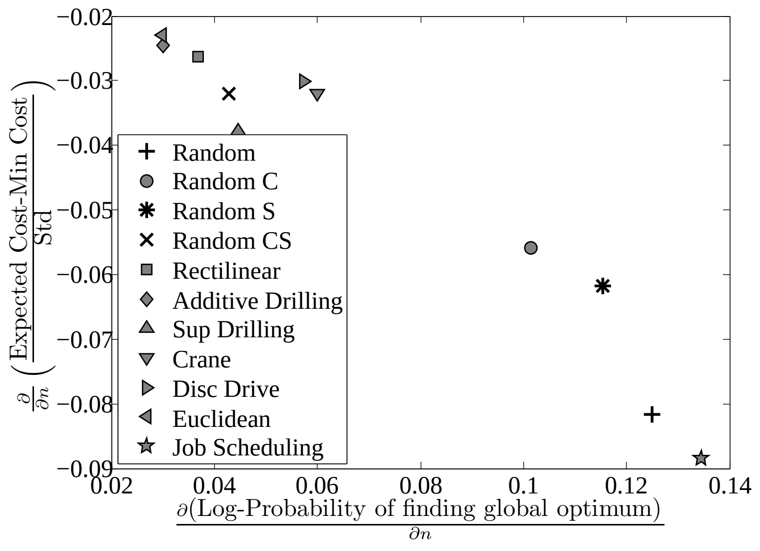


FIGURE 4.21: The increasing rate of the gap between the expected cost and cost of the global optimum with n versus the decay rate of the probability of getting to the global optimum with n. The data are found for n =20 to n =80 with the step of 10 for 20 different problems and 10^4 descents on each.

4.3.9 Distance Between Optima

One important property of the fitness landscape is the distance between the local optima which shows how the local optima are correlated. The distance between two solutions is simply defined as the number of non common edges in the two solutions (note we are using here the Hamming distance rather than the number of 3-opt moves between configurations). For symmetric problems the edges are undirected, so both the directions of the edges are considered the same. The histogram of the distances between the local optima for two different problem types is shown in Figure 4.22. The results are averaged over 20 randomly drawn problem instances for n =50. The interesting property of the histogram is the peaks in the histogram of Additive-Drilling problem, one higher and wider peak at the left, and one lower and thinner at the right side. This peculiar property is because of the nature of this problem. As described in section 4.1.6, in the definition of the Drilling problem, the energy needed for the machine to move the drill down is zero. Due to this property, although this problem is not a symmetric problem, some sub-tours in a solution can be reversed without posing a significant change to the cost of the solution. The change to the cost of a solution posed by reversing a sub-tour is the difference in the y-coordinate of the starting and the ending point of the sub-tour. In this sense the problem can be considered as a partially symmetric problem. This property causes the peculiar behaviour in the histogram of the distances between the local optima.

The histogram of the distances has a bell shape in the Euclidean problem. This bell shape behaviour of the histogram is seen in all the problem types except the Additive-Drilling and in some degrees in the Sup-Drilling problem. In all the problem types, the peak of the histogram is located at the left side of the expected distance between two randomly drawn solutions. Note that for the size of n =50, the expected distance between two randomly drawn solutions is around 49.5. Therefore the histograms of Figure 4.22 clearly suggest that the local optima are much closer to each other than the expected distance between two random solutions.

Figure 4.22 shows the histogram of distances for all local optima irrespective of their cost. It is also interesting to see how the distance between local optima depends on their cost. We show this information for the Sup-Drilling problem in Figure 4.23 where a 2-dimensional histogram is plotted. We observe that the best (lowest cost) local optima are closer to each other than the less fit local optima. Half of the best 10% of local optima are within a distance of 5 from each other. We observe that the less fit local optima are much further from each other than the best fit optima.

4.3.10 Distance to Global Optimum

Another quantity of interest is the distance between the local optimum and the global optima as a function of the cost. To compute this we have divided the local optima into 100 bins according to the log of their cost and computed the minimum distance from the local optima to a global optimum (for TSP there is usually a unique global optimum). In Figure 4.24 we show the mean

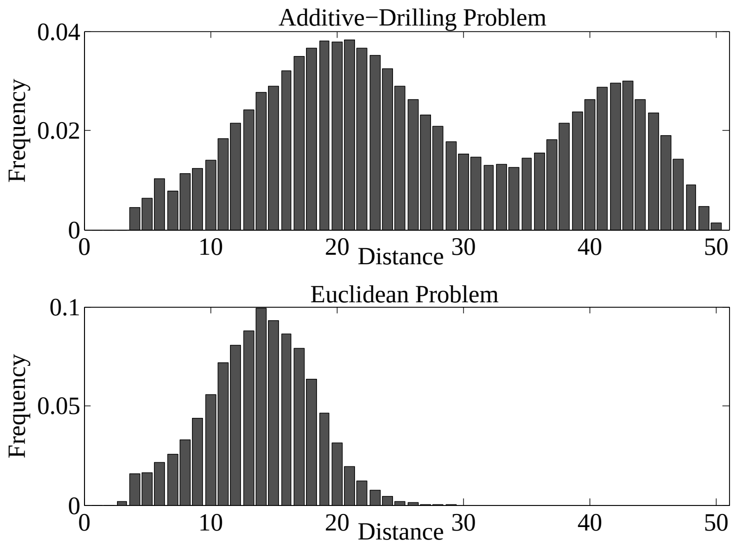


FIGURE 4.22: Histogram of the distances between the local optima for two different problem types. The size of the problem is n =50, and the results are averaged over 20 different randomly drawn problem instances. The number of hill-climbs is 10^5 on each problem instance.

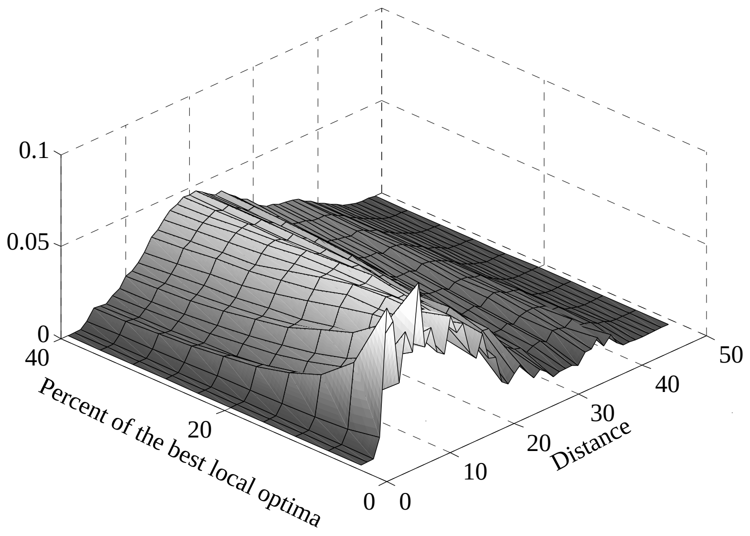


FIGURE 4.23: Histogram of the distances between the best local optima for different number of local optima for Sup-Drilling problem. The size of the problem is n =50. The data are averaged over 20 different problem instances and 10^5 descents on each.

distance to the global optima versus the mean bin cost plotted on a log-log scale. The data is consistent with a straight-line fit suggesting that the mean distance grows in proportion to c^r where c is the cost and r is some rate given the gradient.

There are fluctuations from this behaviour. To measure these, we compute the correlation between the costs in each bin. In Figure 4.25, the gradient and correlation is plotted for different problem types. Among all the problem types, the symmetric random problems show the smallest gradient. The Job-Scheduling and the Sup-Drilling problems show the largest gradient,

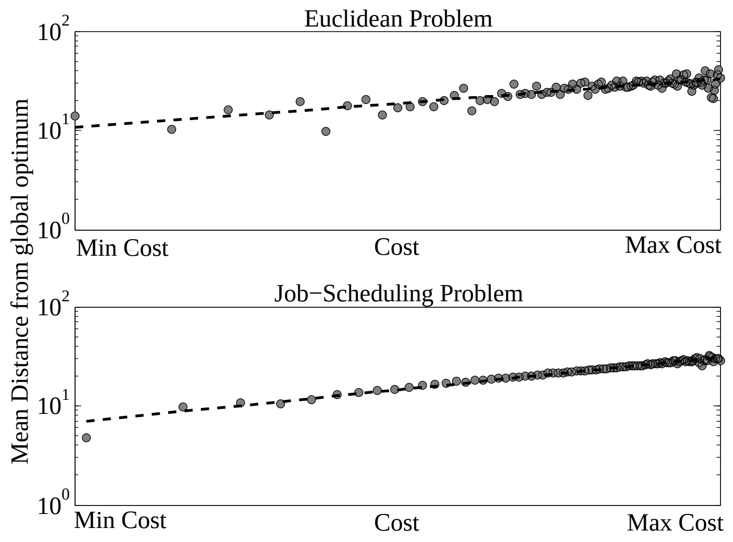


FIGURE 4.24: Measure of the mean distance to the global optimum from a local optimum of cost c on a log-log scale.

indicating that when solving these problems, the selection mechanism moves the population toward the global optimum, at a faster rate. After them are Rectilinear, Additive-Drilling, Crane and Euclidean problems. In terms of correlation the problem types can be divided into two different groups. The Additive-Drilling, Random-CS, Euclidean and Rectilinear problems show a weaker correlation, which means that a population based algorithm has difficulty when trying to estimate the location of the global optimum from the local optima it has currently found. The other problems show a strong correlation which means it is easier for a population based algorithm to find the location of the global optimum through the local optima.

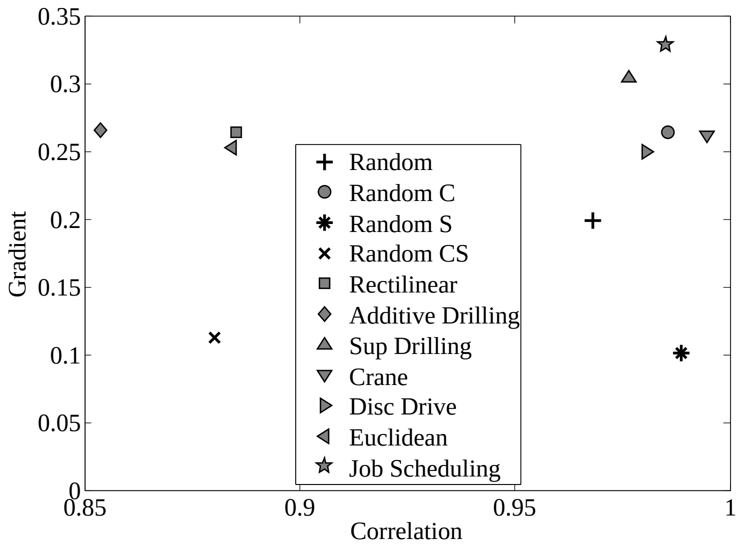


FIGURE 4.25: The gradient of the fitting line in Figure 4.24 versus the correlation for different problem types. The size of the problem is n =50, the data are averaged over 20 different problem instances and 10^5 descents on each.

The data presented in Figure 4.25 is for $n = 50$. The graph shows the same qualitative behaviour for every system size. By rescaling the axes we can collapse the curves for different problem sizes onto a universal curve. This is shown in Figure 4.26 and demonstrates that the distances from a local optimum to a global optimum scales linearly with the problem size. The data in this figure are for the Euclidean problem. The scaling behaviour of the curve is the same for all the problem types. Making the last step from an optimum of cost $c_{min} + \varepsilon$ to c_{min} clearly grows with n, although it is difficult to be sure how this distance grows. This question depends on extrapolating the curve in Figure 4.26 to the y-axis.

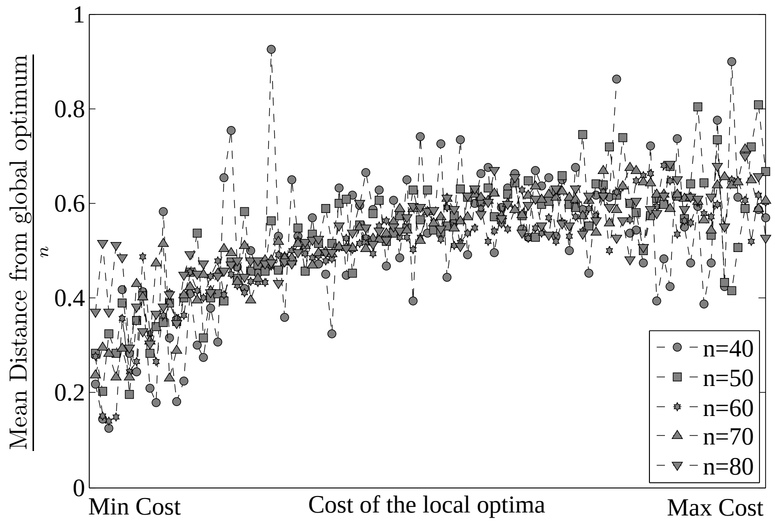


FIGURE 4.26: Measure of the mean distance to the global optimum from a local optimum of cost c for different size of the problem. This is for Euclidean problem, averaged over 20 problem instances.

Another way of viewing this information is as a density plot of the cost versus distance to the global optimum. We show these plots for a Euclidean problem and a Random problem in Figure 4.27. There is clearly a strong fitness distance correlation for the Euclidean problem, but very little for the Random problem. Many population based algorithms attempt to exploit the fitness distance correlation. Clearly, there is some hope in the case of Euclidean problems, but much less hope for Random TSP problems.

4.3.11 Expected Cost in Hamming Sphere

We noted earlier that all the problems had a relatively slow decline in the auto-correlation function, corresponding to a large correlation length. This is a result of the cost being the sum of all the edges in the tour. If we make a local move which changes only a few edges then there will be only a relatively small change in the cost. To examine this we study the mean cost in a Hamming sphere from a configuration. The Hamming sphere of radius h being the set of configurations that differ from the start configuration by exactly h edges.

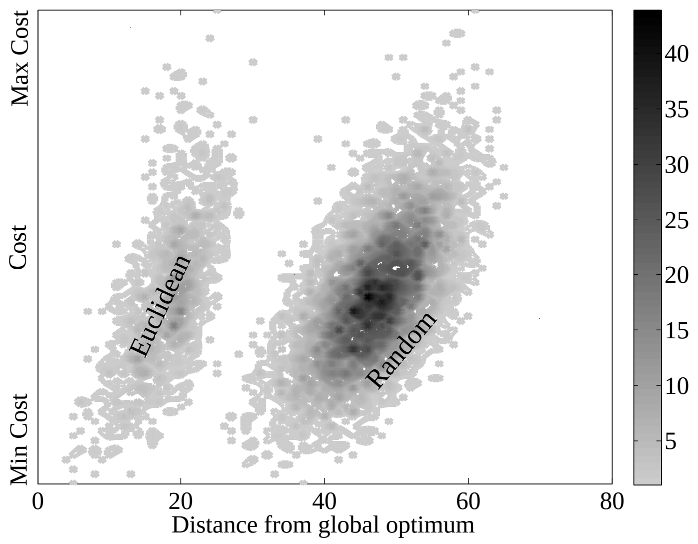


FIGURE 4.27: Density plot of local optima as a function of their distances from the most frequently visited global optimum. The shading of the point shows the number of local optima at that cost and distance. The size of the problem is n =80 and this is for two problem types, Euclidean (left side) and Random (right side) problems. The number of Hill-Climbs is 10^5.

Figure 4.28 shows the expected cost of a configuration in a distance sphere of radius h from a global optimum. To construct the graph, we performed some random number of 3-opt moves to the global optimum and then measured the distance from the global optimum. The process was repeated 10^5 times and the costs at each Hamming distance averaged.

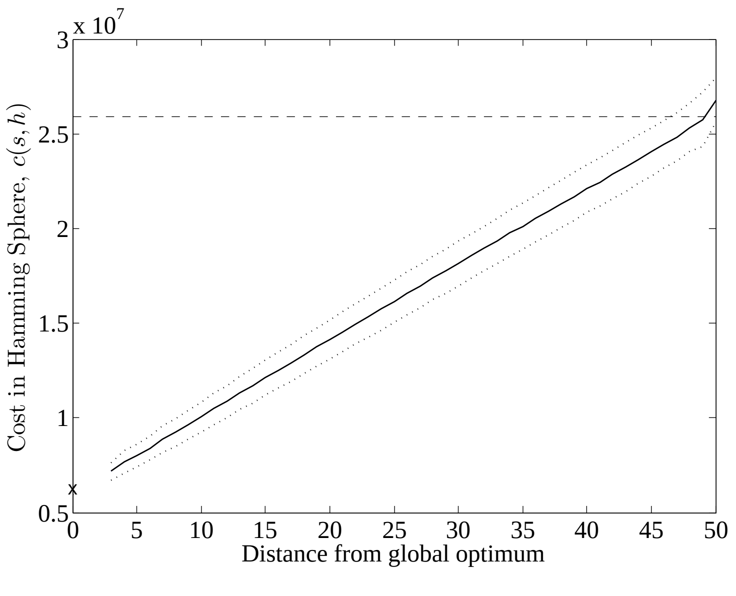


FIGURE 4.28: Expected cost of configurations in distance sphere of radius h around a global optimum for n =50. This is for Euclidean problem. The dotted lines show one standard deviation around the mean. The dashed line shows the expected cost of the random solutions.

Interestingly, we observe that the cost falls off linearly with the distance up to a Hamming distance of $n-1$. We note that the average Hamming distance between random tours is around

$n-1$. We also show the expected cost of a random tour in Figure 4.28 as a horizontal dashed line. A very simple model for the expected cost in a Hamming sphere of radius h about a tour s is

$$c(s,h) \approx \left(1 - \frac{h}{n}\right) c(s) + \frac{h}{n}\bar{c}$$

where $c(s)$ is the cost of the current tour and $\bar{c}$ is the average cost.

4.3.12 Return Probabilities

Another interesting property is the probability of returning to a local optimum starting at a fixed distance from the optimum. In Figure 4.29 we show these return probabilities for four problems starting from the global optimum using 3-opt moves. The curves are not monotonic because 3-opt moves can change some number edges easier than others. For example, it is easier to change 3 edges than 5 edges. As we would expect the return probabilities falls off faster as the problems become harder.

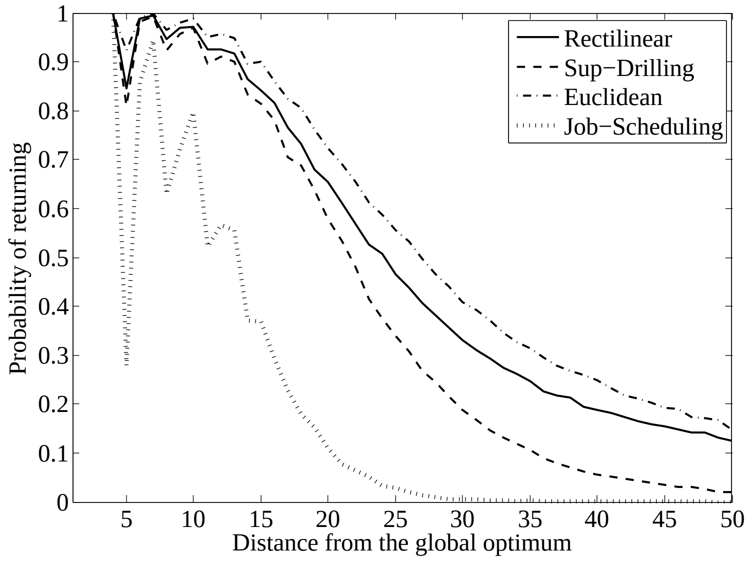


FIGURE 4.29: The return probability starting from different distances from the global optima. This is for four different problem types for size of the problem n =50.

4.3.13 Principal Component Analysis

Although we have seen that the number of local optima grows exponentially with the system size, many of these may be formed by combining other optimal tours. For example, we could have a tour consisting of 10 segments. There may be two routes through the cities in each segment both of which are local optima. In this case there would be $2^{10} = 1024$ local optima which are a consequence of these 10 binary choices. To investigate the underlying variety in the local optima we have used principal component analysis (PCA) to find the subspaces with a large variation. To do this we consider each tour as a binary vector in the space of edges. That is, for a symmetric problem the tour is represented as a binary vector consisting of n non-zero components in a total of $n(n-1)/2$ possible components. Similarly for asymmetric problems the tour is represented as a binary vector consisting of n non-zero components in a total of $n(n-1)$ possible components. To perform PCA we subtract the mean vector of all the optima and then compute the eigenvalues and eigenvectors of the covariance matrix (we can compute this more efficiently using singular value decomposition).

To illustrate this approach we show the average solution of the local optima in Figure 4.30 for a Euclidean problem with 30 cities. The width of the edges shows the number of times they appear in different local optima. It is clear from the graph that there are many edges that do not appear in any of the local optima, and there are some that appear in all the local optima. This property means that the search algorithms do not need to involve such edges in their search process, and a huge proportion of the landscape can be ignored in the search process.

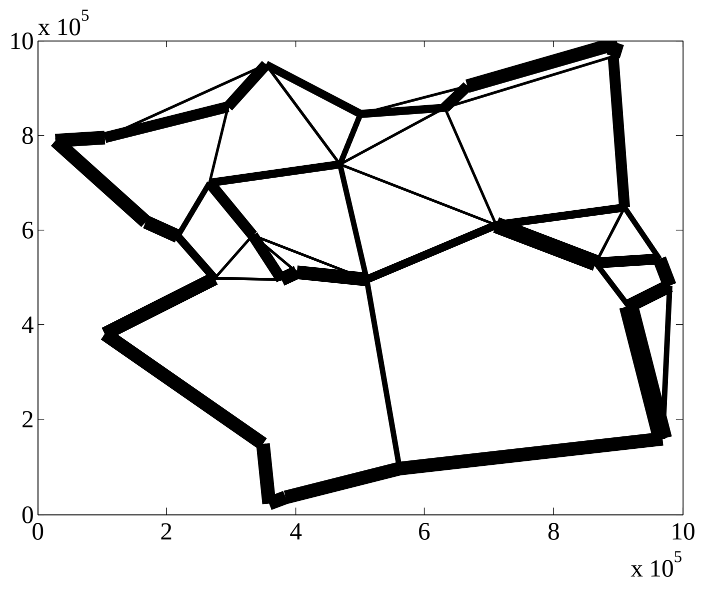


FIGURE 4.30: The schematic shape of the average of local optima solutions for a Euclidean problem with 30 cities. The width of the edges shows the number of times they appear in different local optima.

The eigenvectors correspond to orthogonal directions in edge space. The eigenvectors with larger eigenvalues correspond to directions where there is a large variance. That is, the local optima vary significantly in these directions. We can think of these eigenvectors or "eigentours" as showing the most significant way in which the local optima differ. In Figure 4.31 we show the eigenvector corresponding to the largest eigenvalue (i.e., the principal component) for the same problem shown in Figure 4.30. Some components are positive and others are negative. We can (roughly) interpret an eigentour as a choice between either choosing the positive edges and removing the negative edges or the other way around.

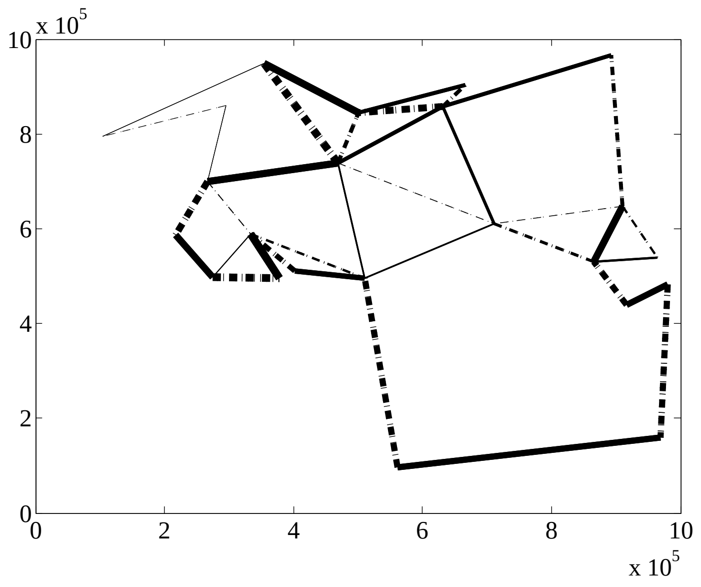


FIGURE 4.31: The schematic shape of the eigenvector of the local optima solutions with highest eigenvalue for a Euclidean problem with 30 cities. The width of the edges shows the values in the eigenvector. The lines represent the positive and the dotted lines represent the negative values.

By plotting the eigenvalues against their rank we can see the dimensionality of the subspace where there is a significant variation between local optima. The size of the eigenvalue is equal to the mean squared reconstruction error of the optima if we project out that dimension. In Figure 4.32, we have plotted the spectrum of eigenvalues for four instances from different problem types. We see that the degree of variation in the Euclidean and the Rectilinear problems we have a relatively small number of non-zero eigenvalues and the variance is small (indicating that the eigentours do not involve that many edges). In contrast, the Scheduling and Sup-Drilling problems show many more directions of variation and considerably larger variations.

One final use of PCA is that it allows us to project the local optimum solution into meaningful low dimensional subspace, allowing some visualisation of the structure of the local optima. We have done this for six problems in Figure 4.33 where we have plotted the local optima into the space spanned by the first two principal eigenvectors (the eigenvectors with the two largest eigenvalues). Furthermore, we have plotted the data so that the size of the dots represents the fitness of the local optima; the better local optima are represented with larger dots.

The first graph is for a Random problem instance, where there is no particular structure in the local optima. For all the problem instances of the Crane problem we almost always see a squared clustered structure for local optima. The squares may form any angle with respect to the horizontal line or may consist of different number of clusters for different problem instances, but the common feature among all the problem types (except Random problems) and problem instances is the fitness of the local optima forming the clusters. We see some of the clusters consisting of better local optima and some other containing inferior local optima solutions. The third graph is for Euclidean problem, of which the projected local optima usually form two

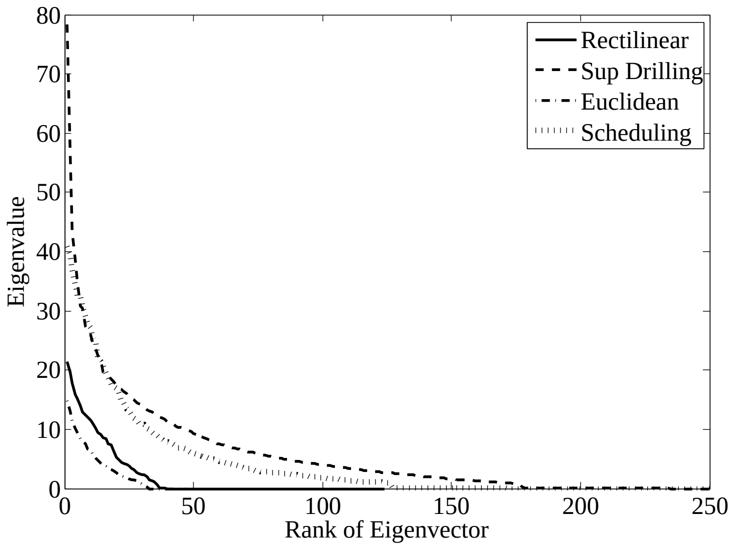


FIGURE 4.32: The eigenvalues of the eigenvectors of the edges in local optima in a descending order. This is for four different problem types for size of the problem n =50.

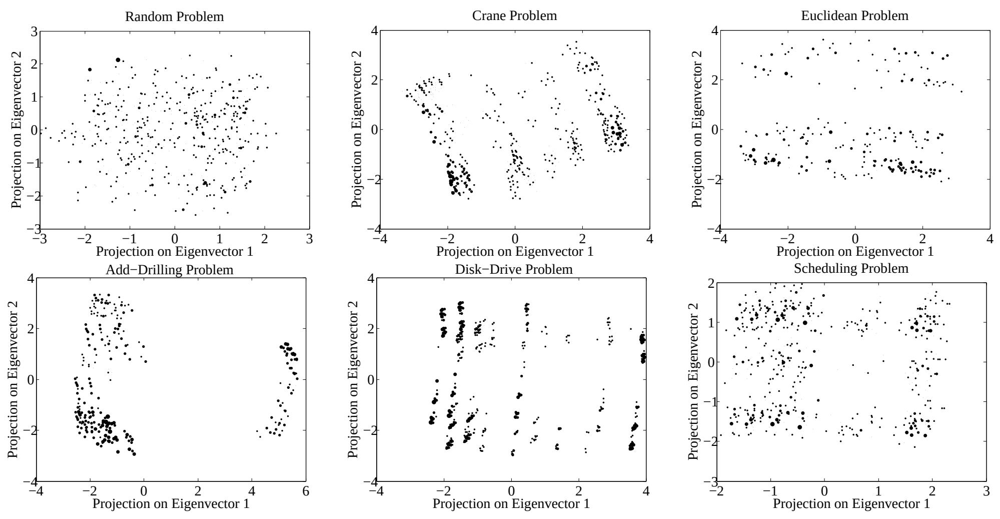


FIGURE 4.33: The projection of the local optima solutions into two dimensions using the two eigenvectors with the highest eigenvalues for three problem types.

parallel elongated football shape clusters. But not all the problem instances show exactly the same behaviour: the angle of the clusters to the horizontal line differs from problem instance to another, the clusters are occasionally not parallel to each other and sometimes there are more than two clusters. For the Additive-Drilling problem we see a similar behaviour to the Euclidean problem, but almost always the number of the clusters is two (in over 50 problem instances we studied) and the gap between the clusters is much wider. The next graph shows the Disk-Drive problem, where the local optima usually form parallel lines of clusters. In this problem type the local optima sometimes form two or more parallel lines and in some cases 10 parallel lines. The lines are sometimes close and mixed together such that they form a square shape. And finally the last graph shows the Job-Scheduling problem, where for almost all the problem instances we observed (for more than 50 problem instances with different sizes), the local optima form squares, although the squares sometimes consist of some clusters.

4.4 Conclusion

For TSP the number of steps to reach a minimum appears to grow linearly for most problem types. However, each step of 3-opt naively takes $O(n^3)$ time to compute (as we potentially have to search all edge triples to find one that can be optimised), this can be time consuming. This does not, in itself, make the problem difficult. What makes the problems difficult is that the local optimum we find is unlikely to be the global optimum.

For all problem types we observed that the number of local optima grows exponentially as some function of the size of the instance, but this rate of this growth varied considerably between problem types.

In most TSP types that we examined the fitter local optima tended to have a significantly larger probability of being found than the less fit local optima. Thus, the number of local optima does not necessarily determine the probability of finding the global optimum.

The probability of reaching a global optimum decreases exponentially with the system size. This property is correlated with a widening gap between the cost of the global optimum and the expected cost of an optimum found by local search, measured in units of the number of standard deviations in the cost of the optima found by local search.

There is a correlation between the fitness of a local optimum and its distance to the global optimum with fitter solutions closer on average to the less fit solutions. This behaviour appears to scale linearly with the problem size. However, this correlation is very dependent on the problem type and the correlation is especially low for Random problems.

The difference between the cost of a reference configuration and the average cost of the set of configuration at a fixed Hamming distance, h, from the reference is almost exactly linearly correlated with h. This is a result of the fact that the objective function consists of a sum of terms, where each term consists of only a few variables. Thus when we change a small number of variables we only produce a small change in the cost. This function is almost exactly linear between the cost of the tour we start from and the expected cost of random tour at a Hamming distance of n. This long-range correlation means that there is a considerable similarity in the long-range structure of the fitness landscape.

Chapter 5

Quadratic Assignment Problem

In this chapter we study the landscape of the Quadratic Assignment problem. We start with problem definition and then move to studying different properties of the fitness landscape.

5.1 Problem Definition

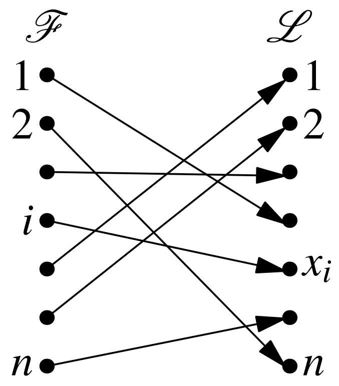
In this section, we describe the quadratic assignment problem and the way we generate the problem instances. We finish this section with a discussion about the local-search algorithm we use to find the local optima.

The quadratic assignment problem is a combinatorial optimisation problem which belongs to the class of NP-Hard problems. Given a set of n facilities, a set of n locations, the distance between every pair of locations, and the flow between every pair of facilities, the problem is defined as finding an assignment of the facilities to the locations which minimises the sum of the distances multiplied by the corresponding flows. Given a set of facilities, $\mathscr{F}$, and a set of locations, $\mathscr{Q}$, a weight function $w : \mathscr{F} \times \mathscr{F} \to \mathbb{R}$ and a distance function $d : \mathscr{Q} \times \mathscr{Q} \to \mathbb{R}$, find $\boldsymbol{x} : \mathscr{F} \to \mathscr{Q}$ such that the cost function

$$c(\boldsymbol{x}) = \sum_{i,j \in \mathscr{F}} w(i,j) d(x_i, x_j) \tag{5.1}$$

is minimised.

We define a configuration as a mapping $\boldsymbol{x}$ shown schematically below,



In this research we study the landscape with respect to a Hamming neighbourhood. That is the distance between two configurations (mappings, $\boldsymbol{x}$) is the number of variables in which the two configurations are different. The local operator on the other chooses two variables and swaps them; therefore, making a move, a variable at a Hamming distance of two is generated.

There are two groups of Quadratic Assignment problems studied in this research, one is the average behaviour random problems, and the second are the real-world benchmark problems from the Quadratic Assignment Problem Library (QAPLIB) found in "http://www.seas.upenn.edu /qaplib/". The real world problems include the typing-time and the frequency of pairs of letters on keyboards, facilities of a hospital and the flow between every pair of them, testing of self-testable sequential circuits and backboard wiring problem. We study a wide variety of properties and try to show how these properties are different between the random and benchmark problems and how they change from one benchmark to another.

To generate random instances of this problem, we have to make two matrices; one is the distance matrix containing the distances between every pair of locations, and one the flow matrix containing the flow between every pair of facilities. To do so we randomly place the locations in a 100×100 square and the flows between the pair of facilities are chosen uniformly randomly between 0 and 100.

5.2 Landscape Analysis

In this section we study some properties of the fitness landscape of the quadratic assignment problem, and show the effect of the size of the problem on the properties of the landscape.

5.2.1 Density of Configurations

We start with considering the density of configurations as a function of the cost. These properties are usually studied when investigating the landscape of combinatorial optimisation problems and we include them here for the sake of completeness.

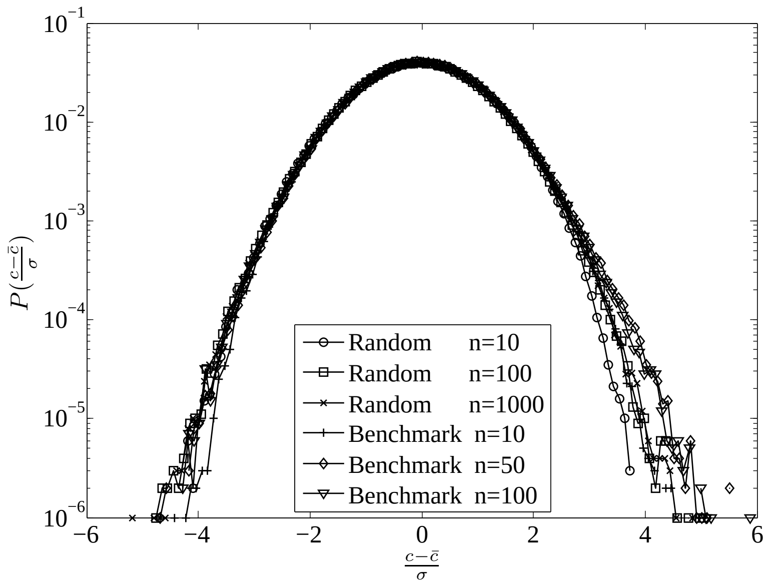


FIGURE 5.1: Histogram of costs for random configurations in particular instances on a logarithmic scale for some random and benchmark problems.

To find the density of configurations, we generate random assignments and compute a histogram of the costs. Figure 5.1 shows the probability of a cost around the average cost scaled by $\frac{c-\bar{c}}{\sigma}$ for different size of the problem for some random and benchmark problems. Regardless of the size of the problem, the results are almost similar for each problem instance. The results are also similar for random or benchmark problems. This means that from the point of view of density of configurations, the real world problems show the same behaviour as the random problems, i.e., the cost of the random configurations is approximately bell shaped around the average cost with a variance of $\Omega(n^2)$.

5.2.2 Auto-Correlation

The auto-correlation of the quadratic assignment problem instances is represented in Figure 5.2 for different random and benchmark problems of different sizes. The rescaling in vertical axis suggests that the auto-correlation falls off exponentially as

$$R(\tau) \approx e^{-\tau/l}, \tag{5.2}$$

where l is known as the correlation length (Stadler, 1996). Empirically, the correlation length is $l \approx 0.28 \times n$. The auto-correlation is taken to be a measure of the landscape ruggedness, the smaller l, the more rugged the landscape.

Based on the scaling used in the horizontal axis, the data suggest that as n increases, the ruggedness decreases. It also indicates that in terms of the landscape ruggedness, the random and real world problems show quite similar behaviours.

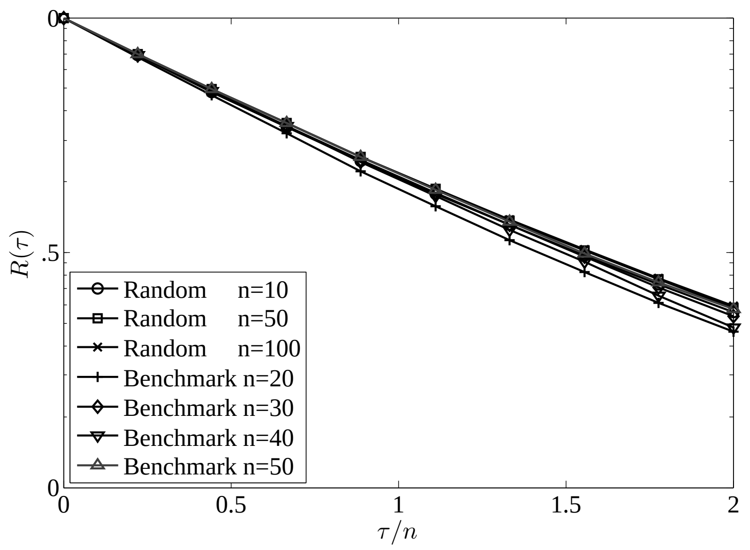


FIGURE 5.2: Auto-correlation for different random and benchmark instances of different sizes plotted against the time difference τ/n. The results for random problems are averaged over 100 runs. The number of steps is 10^8.

Although the auto-correlation of the benchmark problems seems to fall off exponentially with a similar behaviour to the random problems, the correlation length is different for different problems. Figure 5.3 represents the correlation length for random and benchmark problems. The results for the random problems are averaged over 100 problem instances and the result for each benchmark problem is represented with a dot. The error bars show the average correlation length of the random problems plus and minus one standard deviation. The high fluctuations around the average show that the correlation length for the real world problems can vary from one benchmark to another, although it fluctuates in a rather narrow region, between -2 and -4.5.

In order to show what makes the correlation length vary from one benchmark to another, Figure 5.4 shows the proportion of the similar values in the problem matrices (what we call the similarity proportion) versus the correlation length. The presented data suggest that the problems with more similar values in their matrices, have a greater correlation length, which is an indicator of a less rugged landscape. It is not hard to understand why such behaviour happens, having higher ratio of similar values in the problem matrix means that with higher probability, swapping two variables in a solution would not apply a change in the cost; therefore, many of the neighbour solutions tend to have similar values, leading to a less rugged landscape. This property is also reflected in other properties of the fitness landscape which we will discuss in the next sections.

The correlation length describes only local properties of the fitness landscape. As we will see later, the hardness of the problem is related to long-range properties of the fitness landscape.

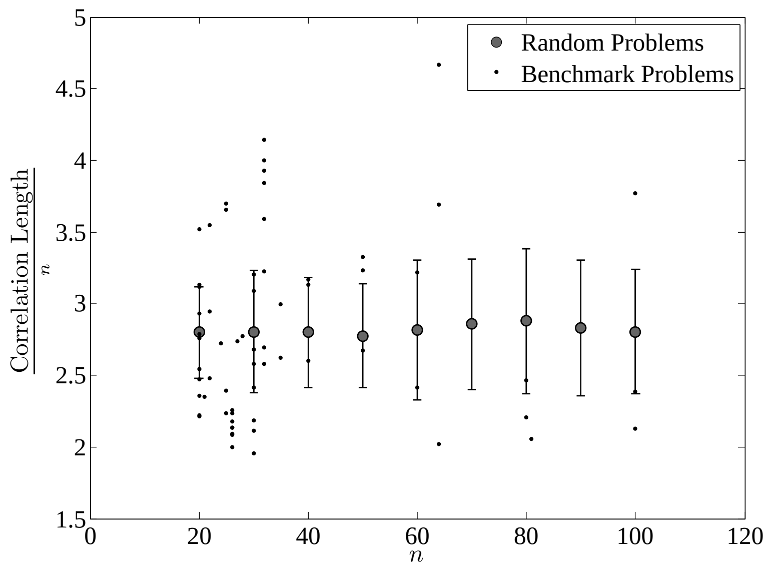


FIGURE 5.3: Correlation length for random and benchmark problems. The results for random problems are averaged over 100 problem instances and the result for each benchmark is shown with a dot. The number of steps is 10^8.

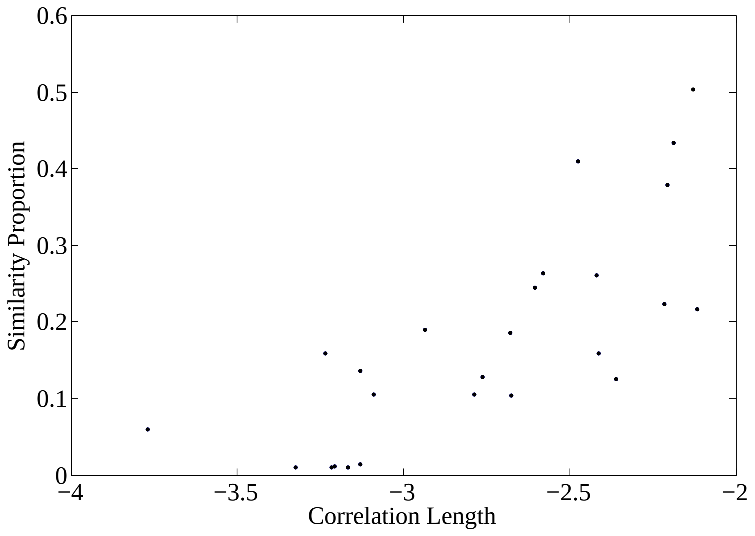


FIGURE 5.4: The similarity proportion versus the correlation length for different benchmark problems with different sizes. The number of steps is 10^8.

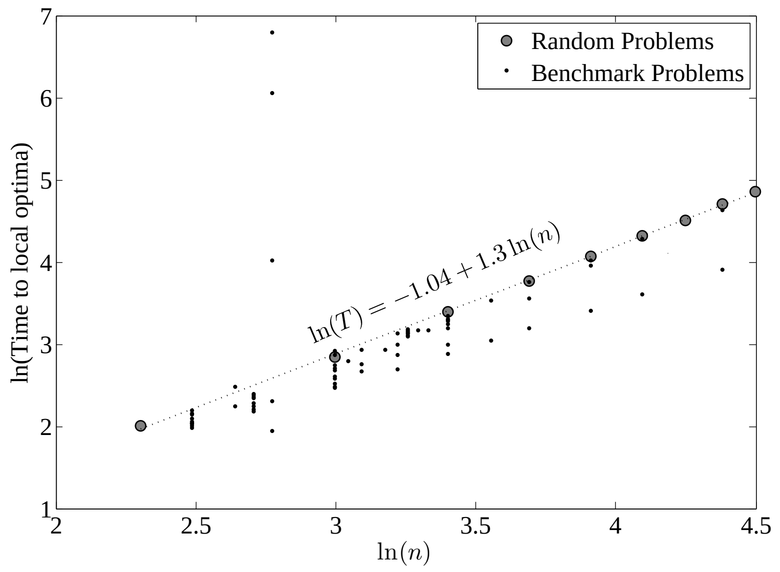


FIGURE 5.5: Time to reach a local optimum versus the size of the problem for random and benchmark problems on a log-log scale. Random data show the average over 100 randomly drawn problem instances and each dot represents a benchmark. The number of runs is 10^5.

5.2.3 Time to Local Optimum

Having described the average behaviour of configurations, we now move on to study properties of local and global optima. These properties are much more relevant to problem difficulty. Our analysis begins by studying the time (number of moves) taken by the local-search algorithm to reach a local optimum. To check if the local-search algorithm has reached a local optimum, the exhaustive local-search algorithm is used, which knows when an optimum is reached.

Figure 5.5 shows the time to reach a local optimum versus the size of the problem for random and benchmark problems. On a log-log scale, the time to local optima for random problems grows linearly with the system size. The data in Figure 5.5 suggest that the time to local optima grows as

$$\bar{T} \approx n^{1.3} \tag{5.3}$$

with the system size.

The data show that in most of the benchmarks, the time to local optima is less than in the random problems. In some benchmarks, the time to local optima is higher than the average behaviour of the benchmarks. A study of the problem matrices of these problems shows that the problems with higher time to local optima have more similar values in their problem matrices. For example, on average 27% of the values in the problem matrix of benchmark problems are similar, while the three benchmarks in Figure 5.5 with a significantly greater time to local optima have more than 60% of the values in their matrices being similar. Although the problems with higher time to local optima have higher ratio of similarity proportion, there are many problems

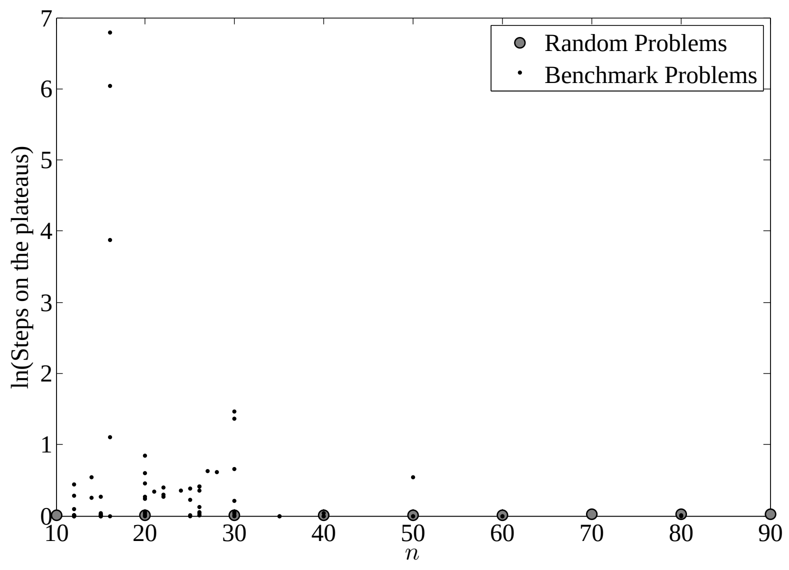


FIGURE 5.6: The average number of steps on the plateaux versus the system size for random and benchmark problems on a logarithmic scale. The results for the random problems are averaged over 100 problems. The number of descents is 10^5.

with high similarity ratio but having relatively smaller time to local optima. This means that having similar values in the problem matrices is a necessary condition for a higher time to local optima, as it makes many of the solutions have the same cost, so many of these solutions may make plateaux in the landscape.

We showed how the time to local optima changes with the system size, but the size of the plateaux on which the local-search algorithm wanders looking for of a cliff (improving moves) is not studied yet. In order to show how the size of the plateaux changes with the system size, Figure 5.6 represents the average number of steps on the plateaux versus the system size both for the random and the benchmark problems. The data for the random problems show that the landscape of the random problems contains almost no plateaux and the number of steps on the plateaux is close to zero for all the problem sizes. The reason for such behaviour is clear, in random problems it is highly unlikely that many of the values in the problem matrix will be similar. Therefore, it is rare to have neighbour solutions with similar costs, the necessary condition for the existence of the plateaux in the landscapes.

Figure 5.7 shows the similarity proportion versus the natural logarithm of the number of steps on the plateaux. The data show that many of the benchmarks have high similarity proportion, but the size of the plateaux is quite small. This means that in many cases a problem may have a large similarity proportion, so having many solutions with the same cost, but the solutions are not in a position to make plateaux in the landscape.

We also study the effect of the similarity proportion on the size of the plateaux in the random problems. In order to generate a random problem with a certain similarity proportion, we first generate a random problem matrix, then we keep choosing random pairs of variables in the matrix and make one of the variables equal to the other one, until reaching the desired

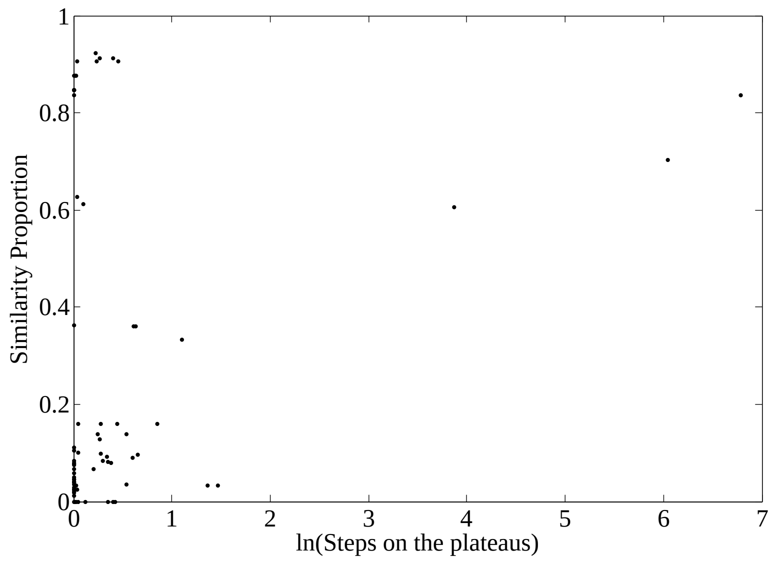


FIGURE 5.7: The similarity proportion the number of steps on the plateaux for the benchmark problems. The number of descents is 10^5.

proportion. Figure 5.8 shows the average number of steps on the plateaux versus the similarity proportion for randomly generated problems. The presented data show that for proportions up to 0.5, the plateaux are small. Having 50% of the values in the problem matrices being similar, one would expect to see large plateaux in the landscape, but that does not happen. This is because although there are many solutions with the same cost in the landscape, many of them are not neighbours, or if they are, do not form plateaux, the same behaviour we saw in some of the benchmark problems. Increasing the proportion from 50% to 60%, the plateaux begin to emerge rapidly. It is clear that for smaller problems the plateaux emerge faster than they do in larger problems. For example for similarity proportion equal to 60%, for $n = 20$, the average number of steps on the plateaux is around 50, while it is about 7 for $n = 30$. Note that in both $n = 20$ and $n = 60$, the similarity proportion is 60%, which means that they have proportionally equal number of solutions with similar cost, but these solutions can form broader plateaux in smaller problems, than they can in larger problems. This may be because in larger problems there are more neighbours to a solution, so a solution with many similar cost neighbours may have a neighbour with a better cost making it a cliff, rather than being a solution on a plateau.

Note however that we have counted the number of steps on the plateaux, not the actual size of the plateaux. We leave the plateaux to a better solution as soon as we find a cliff, so we do not explore the whole plateaux. Nevertheless, the number of steps on the plateaux can be a good proxy for the actual size of the plateaux.

We talked about the time to local optima and the average number of steps on the plateaux, but we did not show the relationship between them. Figure 5.9 shows the number of steps to reach a local optimum versus the number of steps on the plateaux plus one for benchmark problems on a log-log scale. The data suggest that although there is a correlation between the time to local optima and size of the plateaux, there are many benchmarks with no plateaux in their

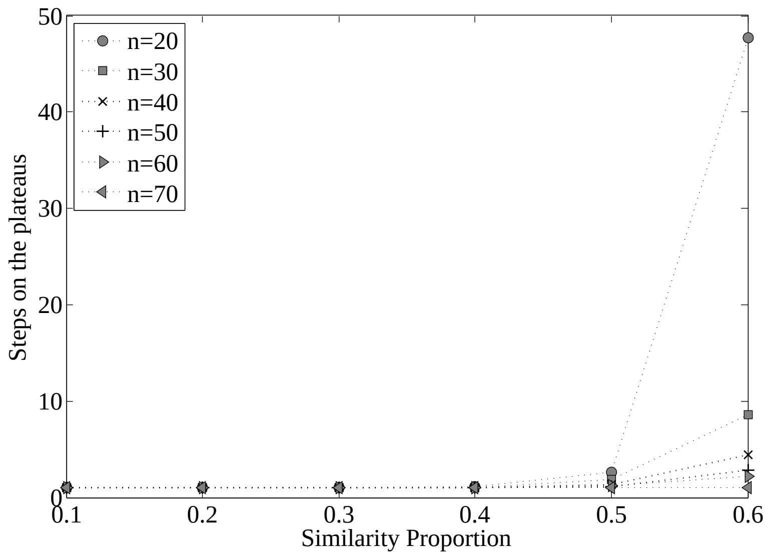


FIGURE 5.8: The average number of steps on the plateaux versus the similarity proportion for random problems. This is averaged over 100 problem instances and the number of descents is 10^5.

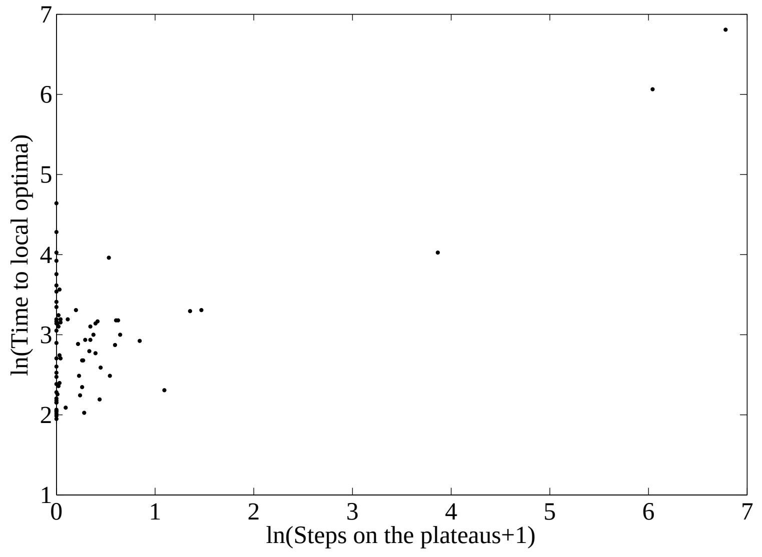


FIGURE 5.9: Number of steps to local optima versus the number of steps on the plateaux for benchmark problems. The number of descents is 10^5.

landscape, while taking many steps for the local search algorithm to get to the local optima in their landscape. Interestingly there are many problems with no plateau in their landscape but showing different time to local optima from one another. This means that the time to local optima does not only depend on the size of the plateaux; it also depends on the number of cliffs (improving moves) the local-search algorithm sees during its search process.

The other important property of the fitness landscape is the number of cliffs the local-search algorithm meets during its search process. We start our analysis by showing the relationship between the number of cliffs in the landscape with the system size. Figure 5.10 shows the

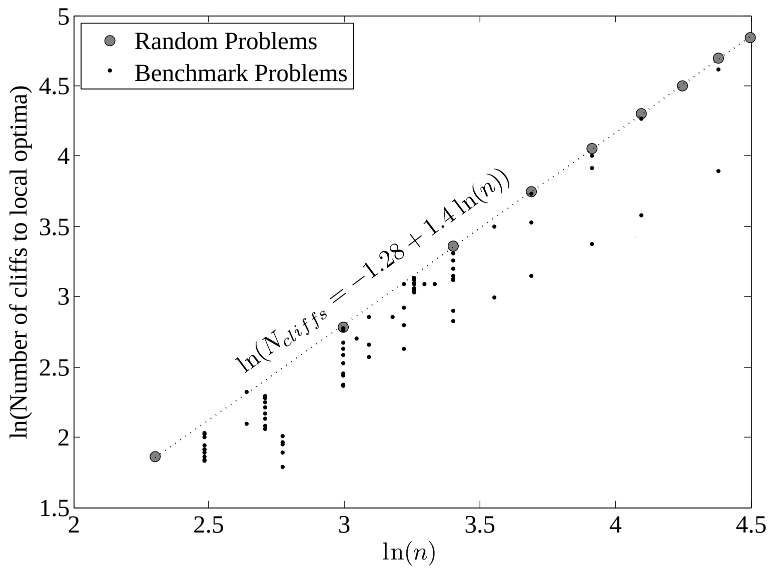


FIGURE 5.10: The number of cliffs the local-search algorithm sees during its search process versus the system size on a log-log scale for random and benchmark problems. The results for the random problems are averaged over 100 problem instances and the number of runs is 10^5.

average number of cliffs the local-search algorithm sees during its search process until it gets to a local optimum. The results for random problems are averaged over 100 problems and the number of runs is 10^5. The data clearly show a polynomial relationship between the number of steps to local optima and the system size. A comparison between Figure 5.10 and Figure 5.5 shows that the number of steps to local optima and the number of cliffs show quite similar behaviour as the system size grows. This means that in the quadratic assignment problem, the number of cliffs has a more important role in time to local optima than the size of the plateaux does. An interesting behaviour is seen in the three benchmarks in Figure 5.5 which show a higher time to local optima than the random problems. These benchmarks are also represented in Figure 5.10 without showing a significantly different behaviour from the random and other benchmark problems. This means that although different real world problems may show different behaviours in the size of the plateaux in their landscapes, they show quite similar behaviour in the number of cliffs the local-search algorithm sees. The other property is that the number of cliffs (improving moves) in the landscape of real world problems is always smaller than that of the random problems. This means that in terms of the time to local optima, the random problems are harder than the real world problems (at least for the benchmarks we have studied and for the way we generate the random problems). However, it is not just the time to local optima which determines the hardness of a problem, the number of local optima and their basins of attraction are other properties which will be studied in the next sections.

The analyses presented so far do not show the size of the plateaux and the height of the cliffs the algorithm sees at each stage of the search process. In order to study this, a density plot of the record of the steps taken by the local-search algorithm to get to the local optima, for 10 000 different descents from random configurations, for $n = 50$, is shown in Figure 5.11. The graph shows the cost of the solutions at each step against the natural logarithm of the number

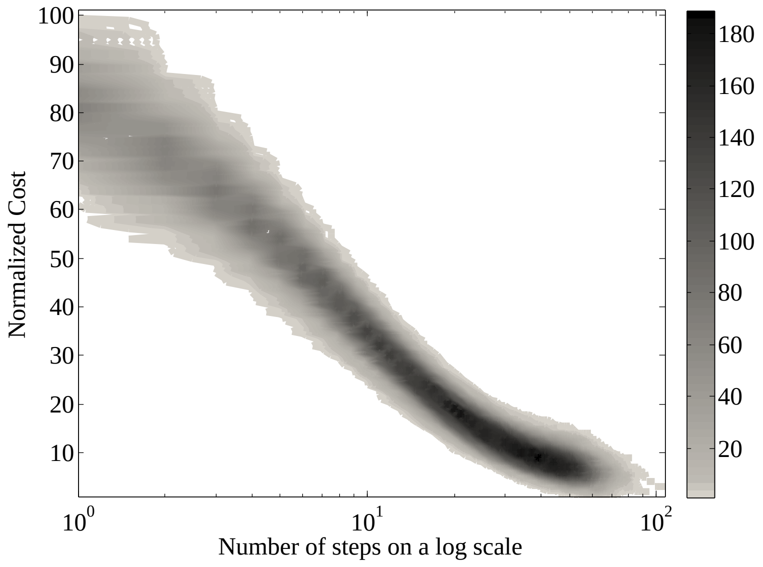


FIGURE 5.11: The density plot of the record of the steps taken by the local search algorithm to get to the local optima for 10^5 different search process, starting from random configurations. The normalised cost of the configurations at each step is represented against the ln number of steps. The size of the problem is $n = 50$. This is for a particular randomly generated problem instance.

of steps. We observe that the shape of the landscape changes as the local-search algorithm gets closer to the local optima. At the beginning, when the search algorithm starts from a random configuration, the algorithm easily finds a lower cost neighbour. Furthermore, the algorithm initially makes a large improvement in cost at each step (note that the algorithm always chooses the best neighbour to move to). However, as the search progresses, not only does the probability of finding a better configuration decrease, but the improvement in cost also decreases. Note that we stop the local-search algorithm when it reaches a local optimum. Thus, the number of steps the algorithm takes on a local optimum to make sure it has reached one is not counted in Figure 5.11.

The data presented in Figure 5.11 show the cliffs and plateaux the local search algorithm sees in solving a random problem. The random problems show similar behaviour; they do not have plateau in their landscape, and the height of the cliffs shrinks exponentially as they get closer to the local optima. Most of the benchmarks show a similar behaviour. This is seen in Figure 5.12, where the density plot of the record of the steps for a benchmark problem with an average behaviour is represented. The graph is similar for most of the benchmark problems.

However, not all the benchmarks show the same behaviour. As we saw in time to local optima, some benchmark problems show plateaux in their landscape. Figure 5.13 shows the density plot for a benchmark problem with huge plateaux. The data show that in most of the runs the search algorithm does not see huge plateaux, although there are times in which the algorithm takes more than 10^4 steps on the plateaux in search of a cliff.

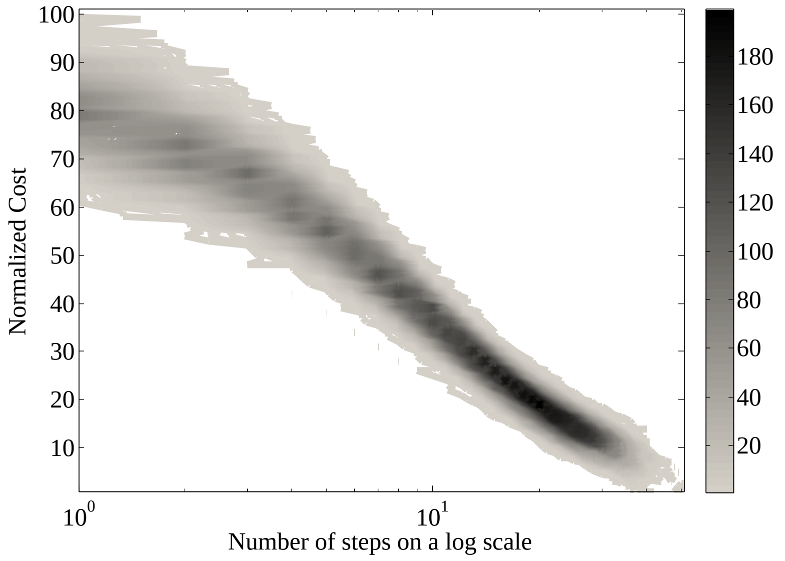


FIGURE 5.12: The density plot of the record of the steps taken by the local search algorithm to get to the local optima for 10^4 different search process, starting from random configurations. The size of the problem is $n = 50$. This is for a particular benchmark with an average behaviour.

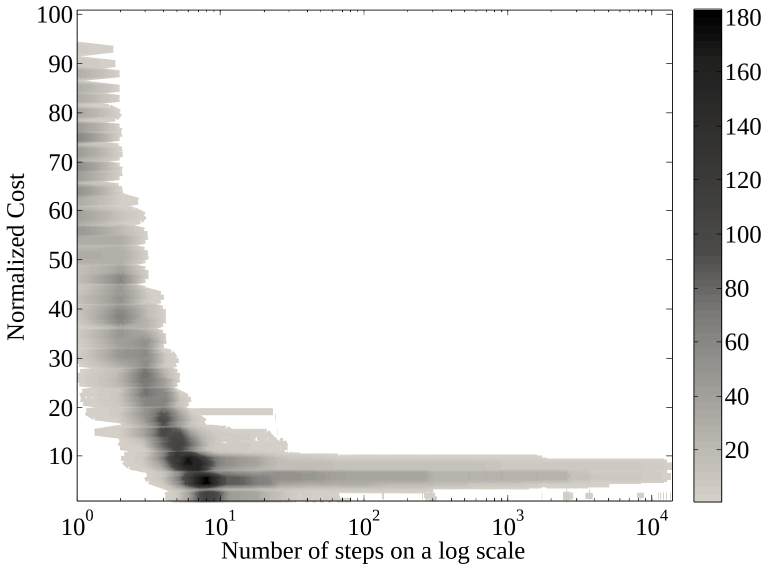


FIGURE 5.13: The density plot of the record of the steps taken by the local search algorithm to get to the local optima for 10^4 different search process, starting from random configurations. The size of the problem is $n = 50$. This is for a particular benchmark which shows huge plateaux in its landscape.

Quite similar to what we saw in Graph-Colouring chapter 3, the graph can be divided into two different parts. In the first part all the runs make steady progress, although the height of the improving moves decreases exponentially as the optimum is approached. The second part (starting after around 10 steps in this case) is due to a small number of runs, where a very large plateau is reached with a cost just above the optimum. When this happens, it can take over three orders of magnitude longer to search this plateau and find the optimum.

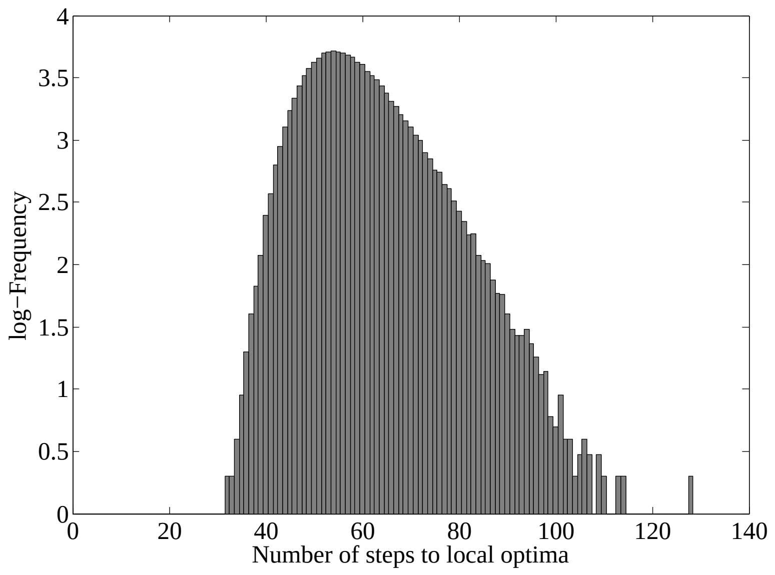


FIGURE 5.14: The histogram of the number of steps to local optima for a random problem instance of size 50 on a logarithmic scale. The number of descents is 10^5.

Figure 5.14 shows the histogram of the number of steps to local optima for a randomly generated problem instance on a logarithmic scale. The histogram shows a roughly bell-shaped (or Gamma distribution) with some rare cases at the right side of the histogram.

Most but not all of the benchmark problems show a similar behaviour as we see in Figure 5.14. Figure 5.15 shows the histogram for one of the benchmarks with plateaux in its landscape on a logarithmic scale. The data show that in most of the cases the local search algorithm does not hit a plateau (or, at least, a huge plateau) and directly reaches a local optimum. However, in this benchmark, there are some rare cases in which the local search algorithm hits a plateau and wanders on it in search of a cliff. Again note that the number of steps taken on a local optimum is not counted in this figure and it shows the actual number of steps on plateaux.

5.2.4 Number of Local Optima

To find the number of local optima in the landscape we use the exhaustive local-search algorithm and store the local optimum it returns. The exhaustive local-search algorithm is repeated a large number of times from randomly chosen starting configurations. Each local optimum with the number of times it is hit is stored. Of course, there is no guarantee that all the local optima in the landscape are found, particularly if a local optimum has a small basin of attraction. Typically there are many local optima with a very small probability of being visited. For example, in one randomly constructed instance with $n = 30$, after 10^5 hill-climbs, there were 85 523 found local optima, of which only 120 were discovered more than 10 times, and 77 536 local optima being found just once. For most of the benchmarks with the size of $n = 30$, all the local optima are hit less than 10 times. For one particular benchmark instance of $n = 30$, again after 10^5 hill-climbs, among 92 418 found local optima, 46 were hit more than 10 times, 92 372 were found less than

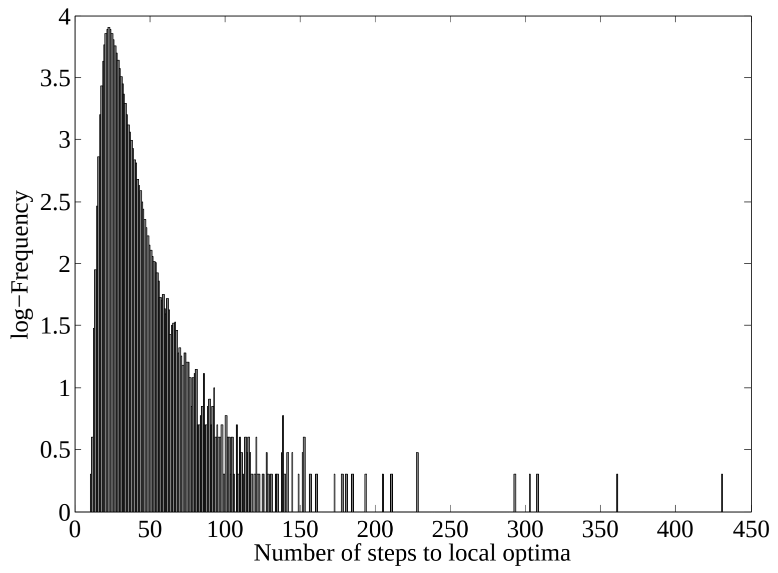


FIGURE 5.15: The histogram of the number of steps to local optima for a benchmark problem of size 30 on a logarithmic scale. The number of descents is 10^5.

10 times, and 87 720 were hit only once. However, those local optima that were observed only once were all high-cost local optima.

The same experiment is performed on a benchmark problem and the results are shown in Figure 5.17. The qualitative behaviour of the curve is quite similar for both the random and benchmark problems. These two similar curves talk about an important property which is seen in all the benchmark and random problems we have studied, not only for the quadratic assignment problem, but also for the MAX-SAT in chapter 2, Graph-Colouring in chapter 3, and Travelling Salesman problems in chapter 4. The property which reflects the size of the basins of attraction of the local optima is that on average, there is a strong correlation between the cost of a local optimum and the size of its basin of attraction. In all the problems, as the cost of the local optima increases, the probability of getting to the local optima (the size of their basin of attraction) decreases exponentially. This property is a desirable one, as it says that through a search process, the better local optima have better chance of being visited, than the high cost ones have. It also, although in an indirect way, suggests the existence of a strong correlation between the cost of the neighbour solutions. Due to this correlation, a better local optimum expands its effect to a longer range, so affecting more distant solutions, and putting them in its basin of attraction. This is akin to the massive object's effect on the space-time fabric in general relativity, the better local optima curve the landscape around them in a stronger way, so more local search process are trapped in their basin of attraction.

Fitting the data for the mean count of the local optima at each cost level in Figure 5.16 to a straight line gives another way to estimate the probability of finding a local optimum. For the case of Figure 5.16, where $n = 30$, the gradient of the fitting line to the data for cost $c = 1.908 \times 10^6$, to $c = 1.913 \times 10^6$, is -0.00014, which suggests that the probability of getting to a particular local optimum at the cost c, decays exponentially as $P_v(c) \propto 10^{-0.00014c}$. (Note

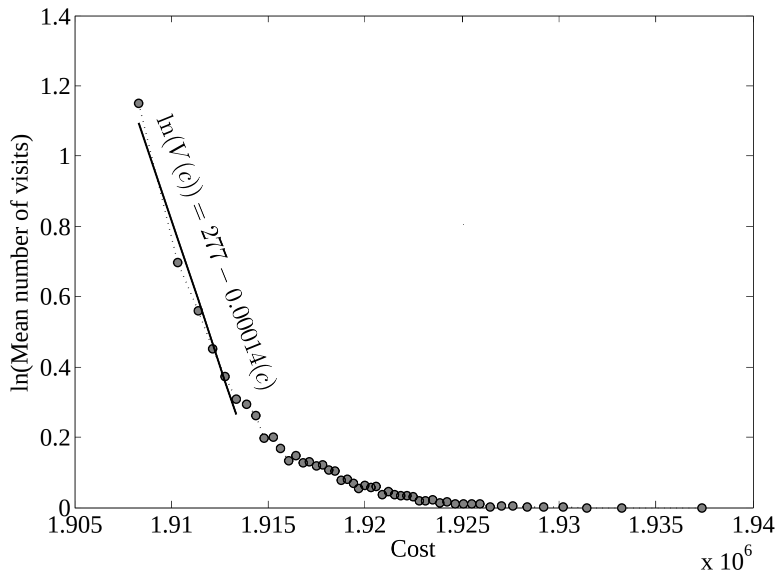


FIGURE 5.16: Natural logarithm of the mean number of times local optima at certain cost are hit in an experiment with 10^5 hill-climbs for a particular randomly drawn instance with $n = 30$.

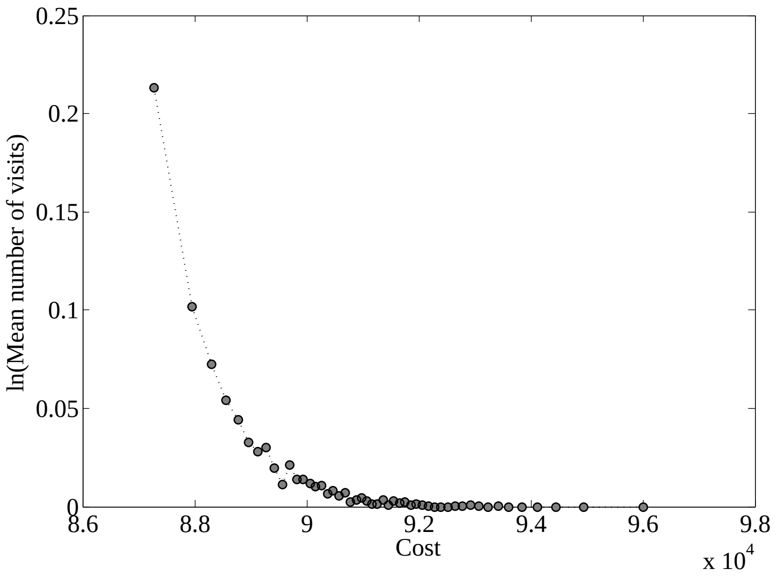


FIGURE 5.17: Natural logarithm of the mean number of times local optima at certain cost are hit in an experiment with 10^5 hill-climbs for a particular benchmark problem with $n = 30$.

that because there are many more local minima at high cost, this does not imply that the search is likely to reach the global optimum). We can give no strong justification for believing that this exponential decay should continue for high costs, although it seems to be a fairly good approximation at low cost for every instance we have examined.

The relationship between the size of the problem and number of local optima is shown in Figure 5.18, where the logarithm of the number of observed local optima divided by $n^{0.25}$ is plotted versus the normalised cost. The results for random instances are averaged over 100 instances. We have performed 10^6 hill-climbs to find the local optima. For large problems we significantly underestimate the actual number of local optima at higher costs (inferior local

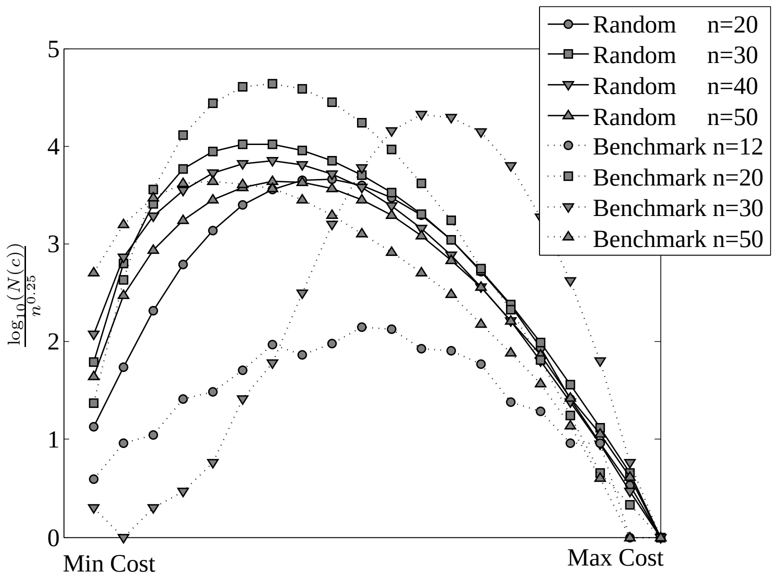


FIGURE 5.18: Natural logarithm of the number of local optima at certain cost for different n of random and benchmark problems. The results for random problems are averaged over 100 instances for 10^5 hill-climbs.

optima). The plot shows that this curve for the random problems is roughly similar for different size of the problem. If it were the case that we had found the majority of local optima, this would indicate that the number of local optima grows approximately exponentially with $n^{0.25}$. The graph also includes some benchmarks of different sizes.

Most of the benchmarks we have studied show similar behaviour to the random problems; the number of local optima grows at a similar rate. In Figure 5.18, the histogram of the local optima at certain cost for some benchmarks represented. The data for the benchmarks of different size show that although the number of local optima seems to grow at the same rate as the random problems, the skewness of the distribution of costs of the local optima is different for different benchmark problems. The number of local optima is also different for different benchmarks.

Note, however, that many of the local optima have been hit just once. This indicates that the number of local optima, particularly for large systems, is underestimated. Although we are aware of this problem, our limited time and computational capacity prevents us from increasing the number of runs to reach better estimation. The other method which provides a more accurate estimation is to sample the configurations for a large number of times and count the number of local optima among them. This method however has a problem, namely the size of the system for which we can count the number of local optima is limited. This is shown in Figure 5.19, where the logarithm of the number of local optima versus the system size is shown on a logarithmic scale.

Figure 5.20 shows the number of local optima in a Hamming sphere around the global optimum on a logarithmic scale. (note that we are aware of the fact that the best observed optimum may not the global optimum; thus, by the global optimum we simply mean the best solution the local search algorithm has found). Although not perfectly, the data become a straight line on

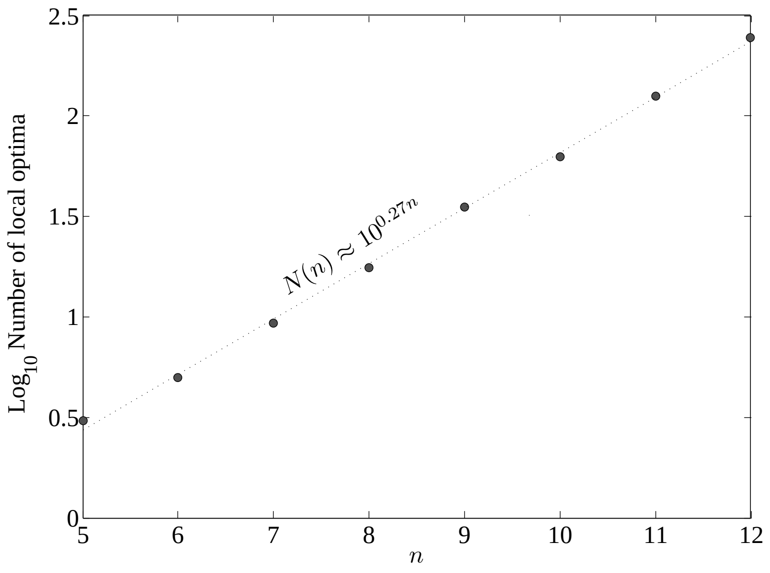


FIGURE 5.19: Logarithm of the number of local optima at certain cost for different *n* of random and benchmark problems. The results are averaged over 100 instances for 10^7 samplings.

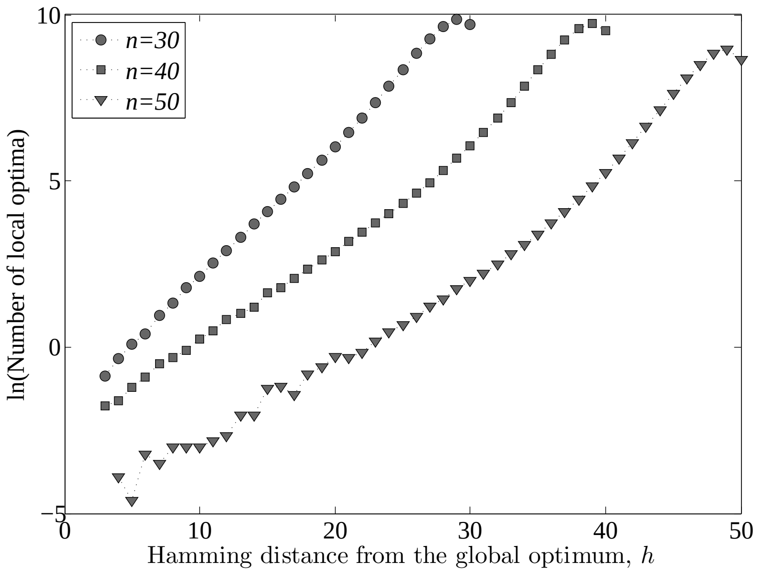


FIGURE 5.20: Natural logarithm of the number of local optima around the global optimum for different *n* of random problems. The results are averaged over 100 instances for 10^5 hill-climbs.

a logarithmic scale. The increasing number of local optima in a Hamming sphere around the global optimum is mainly because of the increase in the number of configurations in a Hamming sphere around it.

To show the number of local optima for different problem benchmarks and compare it with that in the random problems, Figure 5.21 shows an estimation of the number of local optima for different random and benchmark problems of different size. The peak of the number of local optima in Figure 5.18 is taken as an estimation for the number of local optima. Of course such an estimation is unlikely to represent the true number of local optima; however, as many of the

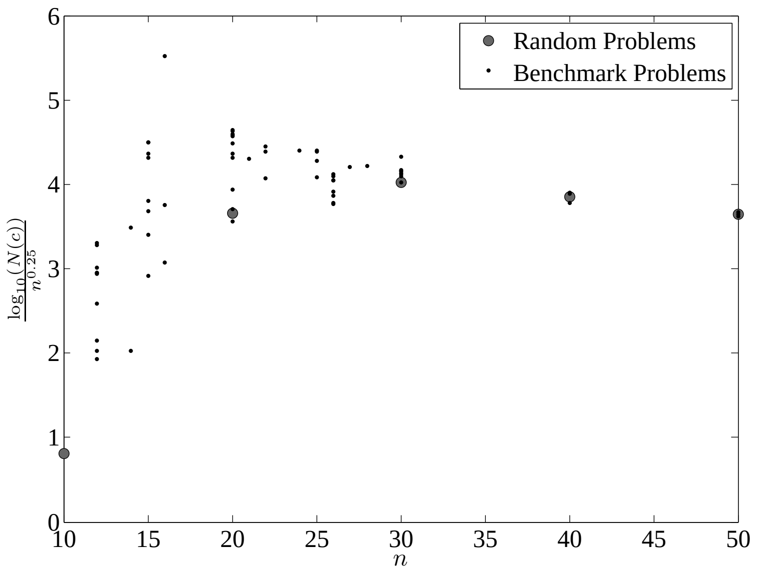


FIGURE 5.21: The number of local optima versus the problem size for different random and benchmark problems. The results for random problems are averaged over 100 instances for 10^5 hill-climbs.

local optima have proportionally small basins of attraction, counting them needs a huge number descents, which is beyond our computational capacity. The data clearly suggest that the number of local optima in benchmark problems is higher than that of the random problems. We saw in Figure 5.10 that the time to local optima for the benchmark problems is less than that of the random problems. In other words, Figure 5.10 says that in terms of time to local optima, the benchmark problems are easier to solve. On the other hand Figure 5.18 says that in terms of the number of local optima, the benchmark problems are the harder ones. This behaviour may represent a property which says that there is a trade-off between the number of steps to local optima and the number of local optima. It may be as the number of steps is greater in the random problems, on average, the local optima have bigger basins of attraction. Since the size of the landscape is limited (n! possible solutions) bigger basins of attraction means fewer local optima.

The difference between the number of steps to local optima and the number of local optima in the benchmark problems and those of the random problems may not be a strong support for the argument we made about the relationship between the number of steps to local optima and the number of local optima. In order to show this relationship, Figure 5.22 shows the number of local optima versus the number of steps to local optima for 67 different benchmark problems of different size. As the number of steps to local optima and the number of local optima both depend on the problem size, we have normalised the number of steps and the number of local optima for each size between 0 and 1; therefore the results for different sizes are comparable. The data clearly suggest an inverse relationship between these two important properties of the landscape. There is some scatter in the data that could be because we do not count the actual number of local optima but use an estimation for that. Or there may be some other reasons for the fluctuations, like there may be some problems with a large number of steps to local optima and

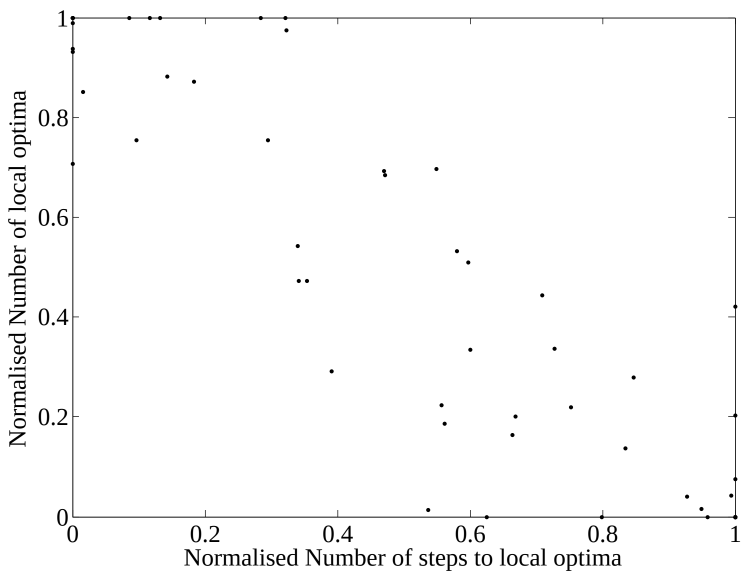


FIGURE 5.22: The normalised number of local optima versus the problem size for different random and benchmark problems. The results for random problems are averaged over 100 instances for 10^5 hill-climbs.

large number of local optima. Note, however, that we are talking about the trade-off between the number of steps and the number of local optima for particular problem size. Of course increasing the size of the problem increases both the number of steps and the number of local optima, but for a particular problem size, having more local optima usually means fewer steps to local optima. The smaller number of steps to local optima means either a smaller number of cliffs, or a smaller size of the plateaux.

Our analysis shows that the distribution of the costs of the local optima has a bell shape around the mean, which means that most of the local optima have a cost similar to the average cost. We also showed that the better local optima have bigger basins of attraction and thus higher chance of being found by local search algorithms. A consequence of this property is that there is clearly a strong bias towards better local optima. Figures 5.23 and 5.24 show the proportion of the local optima at each cost level and the probability of finding local optima at each cost level for two random and two benchmark problems. Note that the data show the empirically measured proportion of local optima at each cost level, and thus underestimate the true proportion of local optima at high cost levels. Figure 5.23 shows that the bias towards the better local optima changes from a problem instance to another. We show two problem instances with different behaviours, one shows a weak, and the other shows a stronger, bias. This means that in terms of finding the better solutions, some problems are easier than others. The same is shown in Figure 5.24 for two benchmark problems of size $n = 20$. In one of the benchmarks the bias is too small, and the other benchmark shows much stronger bias towards the better local optima. This shows how the hardness of the problems (in the sense of finding better solutions) can change from one benchmark to another of the same size.

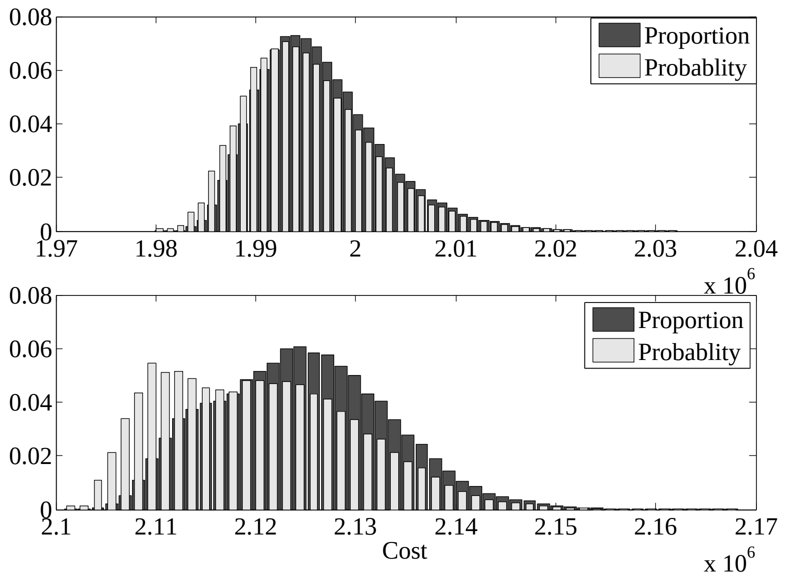


FIGURE 5.23: Histogram of the cost of the local optima and the probability of reaching each local optimum at a particular cost. This is for two random instances of size $n = 30$.

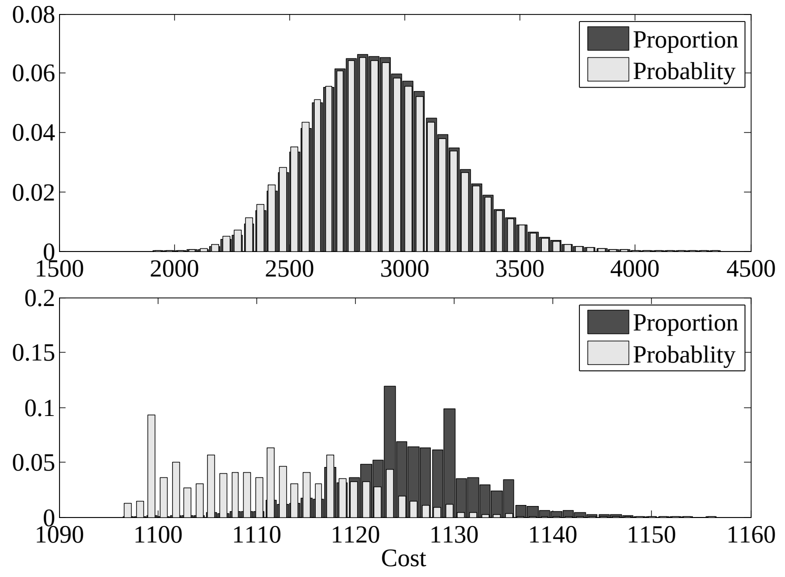


FIGURE 5.24: Histogram of the cost of the local optima and the probability of reaching each local optimum at a particular cost. This is for two benchmark instances of size $n = 20$.

Figure 5.25 shows the bias towards the better local optima versus system size for random and benchmark problems. We use the following measure for the bias,

$$\text{Bias} = \frac{\bar{c} - \mathbb{E}(c)}{\sigma_c}, \tag{5.4}$$

where $\bar{c}$ is the average cost of the local optima, $\mathbb{E}(c)$ is the expected cost of the found local optima by the local search algorithm and σ_c is the standard deviation in the cost of the local optima.

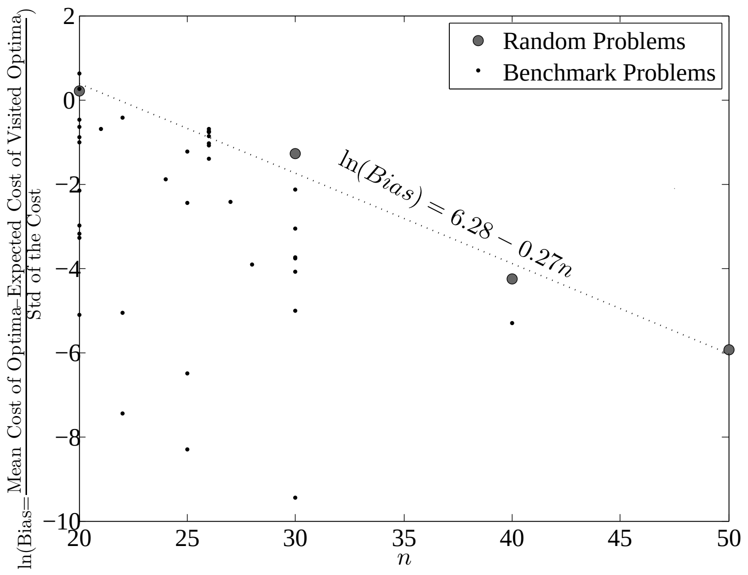


FIGURE 5.25: The bias toward the better local optima for random and benchmark problems versus the system size n on a logarithmic scale. The results for the benchmark problems are averaged over 50 problems, and the number of runs is 10^5 descents.

The data show that as the system size grows, the bias decreases exponentially as

$$\text{Bias} \approx e^{-0.27n}. \tag{5.5}$$

This means that as the size of the problems grows, the chance of getting to the better local optima decreases exponentially. The data also show that the bias of the benchmark problems is smaller than that of the random problems, meaning on average, the local search algorithm has higher chance of finding the better local optima in random problems than the benchmark problems. From the point of view of the probability of finding the better local optima, the benchmark problems are harder problems than the random ones. It also seems that the bias for the benchmark problems shrinks exponentially with the system size.

In order to show the relationship between the bias and the other important property of the fitness landscape, Figure 5.26 shows the bias versus the probability of finding the global optimum for the benchmark problems on a log-log scale. There is a strong relationship between the bias and this probability. This property and what we see in Figure 5.25 show how the probability of finding the global optimum decreases as the system size grows. More detailed studies on the probability of finding the global optimum is described in the next section.

5.2.5 Reaching the Global Optima

As shown in previous sections, from the perspective of finding good solutions, it is a desirable property of many of the combinatorial optimisation problems including the MAX-SAT, Graph-Colouring, Travelling Salesman and Quadratic Assignment problems that the fitter local optima

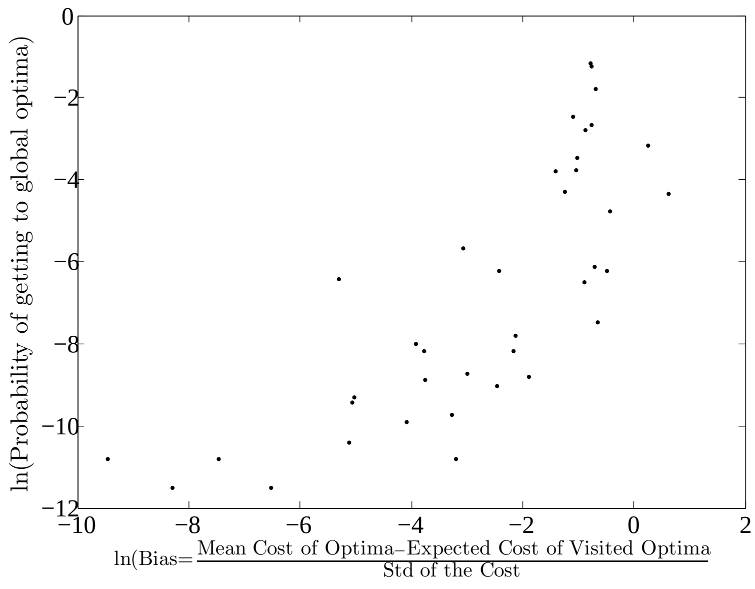


FIGURE 5.26: The probability of getting to the global optima versus the bias toward the better local optima for benchmark problems with different sizes on a log-log scale. The number of runs is 10^5.

have, in expectation, larger basins of attraction. However, the gap between the expected cost found and the cost of the globally optimal solutions grows as the system size, n grows. This is illustrated in Figure 5.27, where the gap between the minimum cost (average cost of the global optima over 100 problem instances) and the mean cost of the local optima found by the local-search algorithm over an ensemble of randomly drawn instances are shown. The gap is defined as

$$\text{Gap} = \left(\frac{c_{min} - \mathbb{E}(c)}{\sigma}\right)^2, \tag{5.6}$$

where c_{min} is the cost of the best found optimum, $\mathbb{E}(c)$ is the expected cost of the found local optima and σ is the standard deviation in the cost of the found local optima. The gap is an indicator of the hardness of the problem, as the wider the gap, the lower the probability of finding the global optimum. The data show that the gap grows linearly with the system size. It is clear that the gap for some of the benchmarks is higher, and for some of the them lower than that in the random problems.

For a fixed instance, the cost of the optima found by the exhaustive search algorithm fluctuates on every run. Figure 5.28 shows the standard deviation in the cost found by different runs of local search divided by $\bar{c}$, plotted against n on a log-log scale. The data suggest that the standard deviation decreases almost linearly with the system size. Since the gap between global minimum cost and the cost found by local search does not decrease with n, then as n grows, finding a global optimum becomes more and more unlikely. The analysis shows that the standard deviation of the cost of the local optima is higher in real-world problems than it is in random problems. In this sense, the random problems are the harder ones to manage.

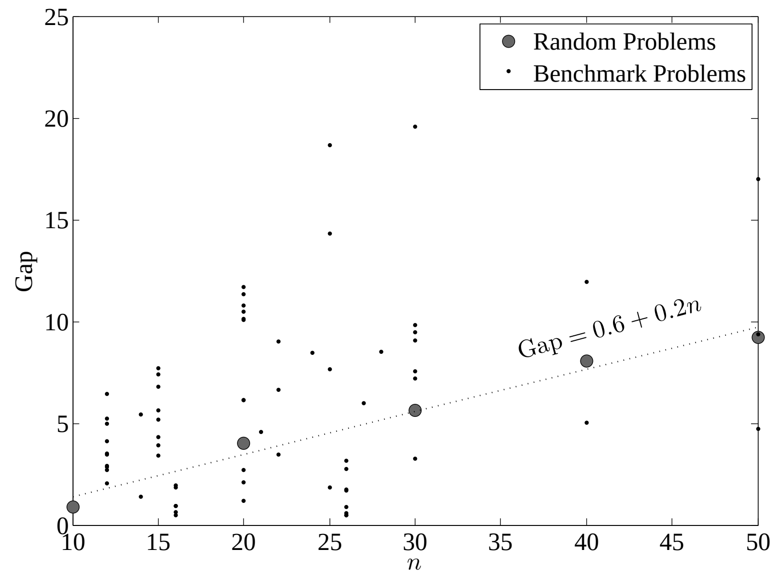


FIGURE 5.27: The gap between the cost of the global optimum and the expected cost of the found local optima by the local search algorithm. The results for the random problems are averaged over 100 problems and each dot represents one benchmark problem. The number of descents is 10^6 runs.

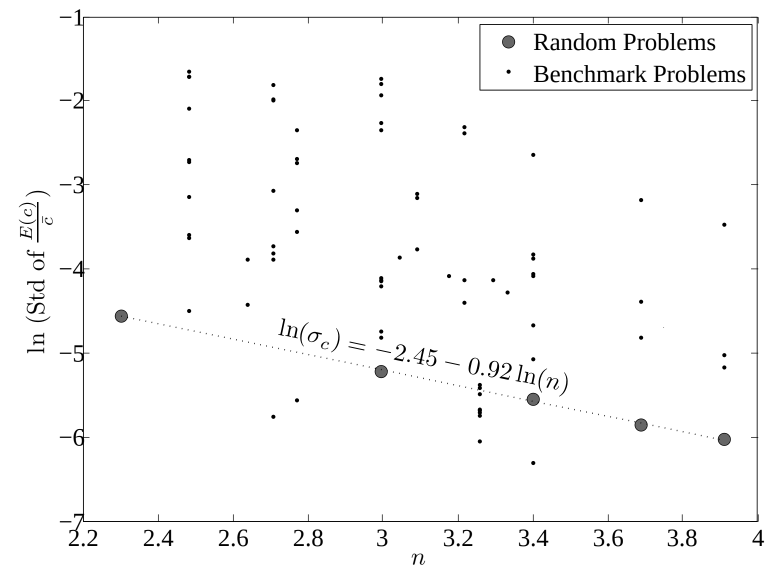


FIGURE 5.28: Plot of the variance of $\mathbb{E}(c)/\bar{c}$ found by local search algorithm. The number of hill-climbs is 10^5 on 500 randomly generated instances.

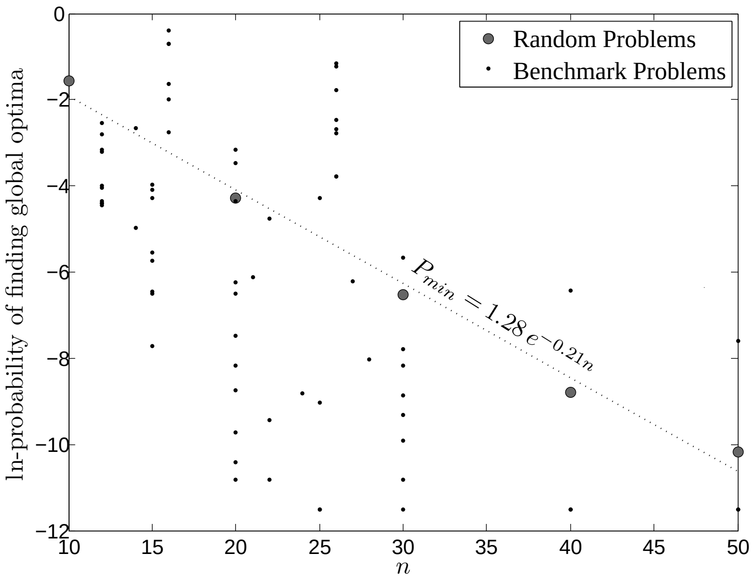


FIGURE 5.29: Natural log-probability of finding the global optima versus n for random and benchmark problems, the number of hill-climbs is 10^5 and the data are averaged over 100 randomly generated instances.

In Figure 5.29, the natural logarithm of the probability of finding a global optimum is plotted against n for random and benchmark problems. The straight line fit is consistent with the hypothesis that finding a global optimum using the local-search algorithm becomes exponentially unlikely as the system size grows. Using the straight line fit in Figure 5.29 to extrapolate to large n, the probability of the local-search algorithm finding a global optimum for an instance of size 1000 would be 8.04×10^{-92}. Although such an extrapolation is unlikely to provide a precise value, nevertheless, it gives a strong indication that for larger problem instances, using multiple runs of local-search algorithms begins to be useless, unless we run it for exponentially large number of times (in the system size). In terms of the probability of finding the global optimum, for the case of real-world problems, it seems that this probability decreases exponentially with the system size.

5.2.6 Distance Between Optima

Another important property of the fitness landscape of the optimisation problems is the distance between the optima, which shows how the optimal solutions are correlated in the landscape. Figure 5.31-a shows the histogram of the distances between local optima, averaged over 100 instances for $n = 50$. The histogram shows that the local optima can be quite far from each other; however, as the expected Hamming distance between random configurations of size $n = 50$ is 49, the data indicate that the average Hamming distance between the local optima is 45.08, which means that the local optima are closer to one another than the average distance between the random configurations. A comparison between the histogram of the distances of all the local optima in Figure 5.30-a and that of the fittest 10% of local optima shows that the histogram of

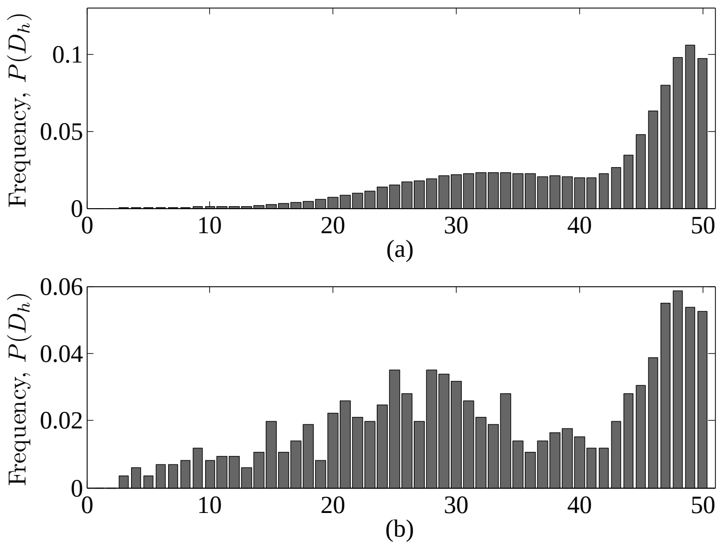


FIGURE 5.30: Histogram of the Hamming distances between a) all the local optima, b) top 10% local optima, for 100 random quadratic assignment problem instances with $n = 50$. The number of runs is 10^5.

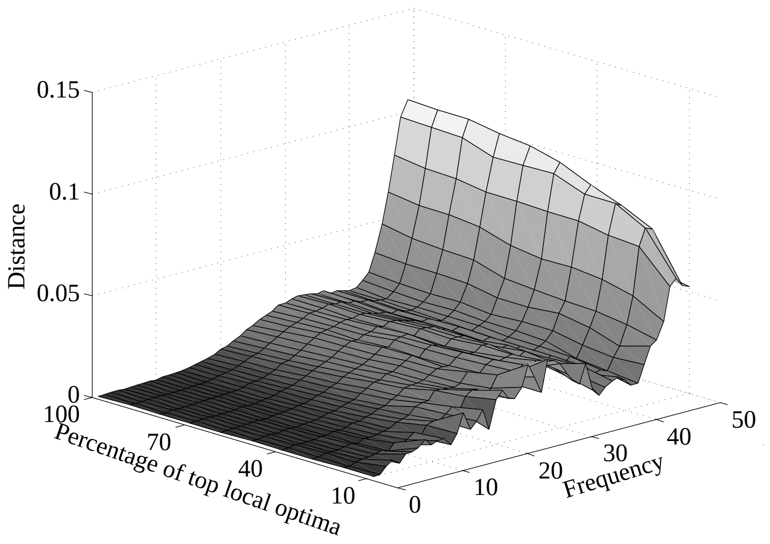


FIGURE 5.31: Histogram of the Hamming distances between the local optima, for different percentage of top local optima for 100 randomly drawn quadratic assignment problem instances with $n = 50$. The number of runs is 10^5.

the better local optima is shifted to the left. This clearly suggests that the better local optima are gathered together in the landscape.

The histogram of the distances between the local optima for different percentage of the top local optima is represented in Figure 5.31. As the inferior local optima are excluded from the histogram, the histogram gradually inclines to the left side and changes its shape from Figure 5.30-a, to the shape in Figure 5.30-b.

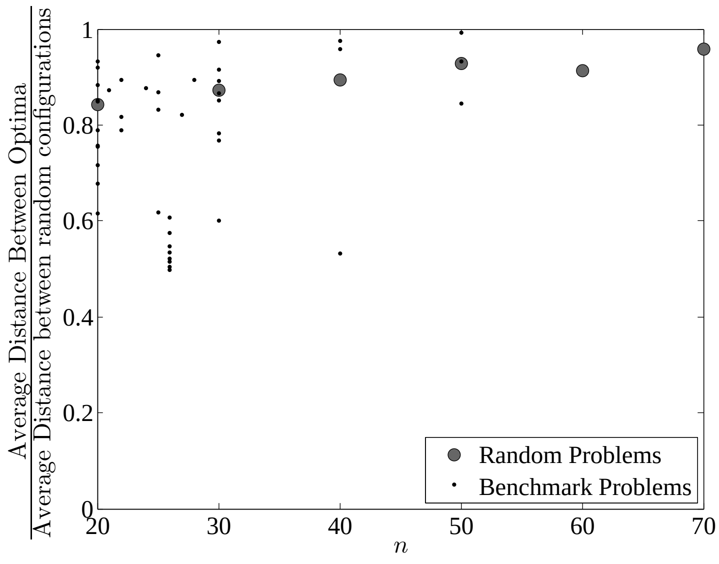


FIGURE 5.32: The average distance between the local optima divided by the average distance between the random configurations $(n-1)$ versus the system size. The results for the random problems are averaged over 100 randomly drawn problem instances and each dot represents a benchmark problem.

The average Hamming distance between the random configurations in quadratic assignment problem is $n-1$ which we showed is greater than the average distance between the local optima. Figure 5.32 shows the average distance between the local optima divided by $n-1$ for different random and benchmark problems of different sizes. The results for random problems are averaged over 100 randomly drawn problem instances. The data suggest that as the system size grows, the average distance between the local optima gets closer to the average distance of random configurations. This means that the local optima are relatively less correlated in larger systems, than they are in the smaller systems. On the other hand, different real-world problems show different behaviours. Some show strong correlation between their local optima and some show almost no correlation; however, the benchmarks exhibit the same behaviour as the random problems. In general, as the system size grows, the local optima show less correlation. This property may seem to say that as the correlation between the local optima decreases with the system size, it gradually becomes impossible for the population based algorithms to exploit such a correlation. However, the fact that the decrease in correlation is because of the proliferation of the uncorrelated inferior local optima, and that it is the correlated better local optima which have better chance of being found by search algorithms (see Figure 5.31), indicate that the population based algorithms still have chance of using this correlation for the larger system size.

Figure 5.33 shows the mean minimum distance from a configuration in a local optimum to the global optimum on a log-log scale for different size of the problem. These results are averaged over all the local optima found in 100 instances. The data, for larger systems, almost fit a straight line on a log-log scale which suggests that as the cost of the local optima grows, the average distance to the global optimum grows as $O((c-c_{min})^g)$, where $g<1$ is the gradient of the fitting line to the data. This behaviour clearly indicates that during the search process, when

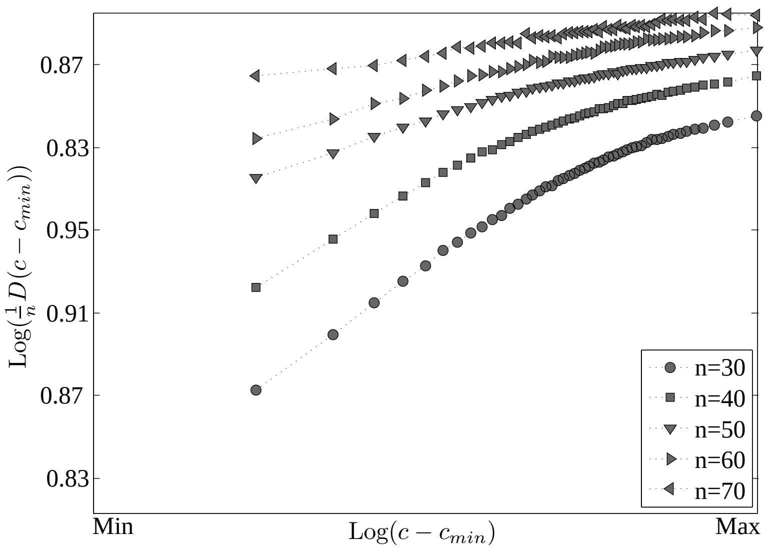


FIGURE 5.33: The mean minimum distance from a configuration in a local optimum to the global optimum on a log-log scale for different size of the problem.

a search algorithm jumps from a local optimum of cost c to a better optimum of cost c' ($c > c'$), the expected distance of the search algorithm to the global optimum decreases by $O((c-c')^g)$. The gradient of the fitting line in Figure 5.33 is less than one, which means that going from a local optimum of cost c to a better optimum of cost c', the local search algorithms make bigger jumps, if they are at a lower cost local optimum. This is similar to what we saw in Figure 5.11, where the lower the cost of the solutions, the smaller the improving moves. The difference is that here in Figure 5.33, we are talking about the local optima, not the neighbour solutions. Note that this is bad news, as getting closer to the global optimum, the search algorithm has to take bigger steps to reach a better local optimum. The biggest step will accordingly be the last step from the closest local optimum to the global optimum.

There are two important properties of the fitness landscape which are reflected in this figure. First as n grows, the distance between the closest local optimum to the global optimum grows faster than linearly. For example for $n = 30$, on average the last step from the closest local optimum to the global optimum is $0.82n$, while for $n = 70$, this distance is $0.94n$. This means that as n grows, the number of configurations in between the global optimum and its closest local optimum grows as $O(n!)$. The other property is the decreasing rate of the distance to the global optimum, as the search algorithm meets lower cost local optima. The data suggest that for the smaller problems, to get to a better local optimum, the search algorithm has to take longer steps (in the system size). This property, in contrast with the first one described, means that as the system size grows, in order for the search algorithm to find a better local optimum, it needs to take smaller steps (in the system size).

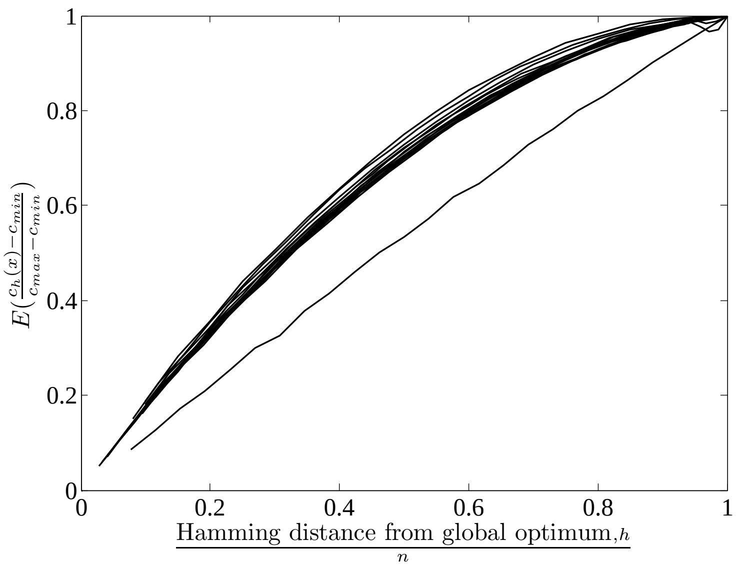


FIGURE 5.34: The normalised expected cost of configurations in Hamming sphere of radius h around a global optimum for different size of the problem and different benchmarks.

5.3 Landscape Correlation

The experiments we have presented so far provide good reason for believing that the quadratic assignment problem is difficult for a hill-climber to navigate. However, as we show in Section 6.1.1, there exist significant long-range correlations in the landscape of the quadratic assignment problem. This is a consequence of the structure of the fitness function. The change in the cost of a configuration due to swapping the assignment of two variables will be equal to the sum of the flows times the distances of the new location of the swapped facilities to all the other facilities, minus the sum of the flows times the distances of the old location of the swapped facilities to all the other facilities. Typically, this change in fitness (which is of order $O(n)$) is rather small, in comparison with the overall cost (which is of order $O(n^2)$). As a consequence, the configurations around a fit configuration will also tend to be fit. This property is also reflected in Figure 5.11, where the density plot of the record of the steps to local optima is represented.

5.3.1 Expected Cost in Hamming Sphere

In Figure 5.34, the expected cost of a configuration in a Hamming sphere of radius h from a global optimum is shown. This is for problems of different size and different benchmark problems. Based on the rescaling used in this figure, the expected cost of configurations in Hamming sphere is quite generic for almost all the problems presented in this figure. The expected cost seems to be growing polynomially with the Hamming distance for most of the problem instances, including the random and some of the benchmark problems. For one of the benchmarks, the expected cost seems to grow linearly with the Hamming distance.

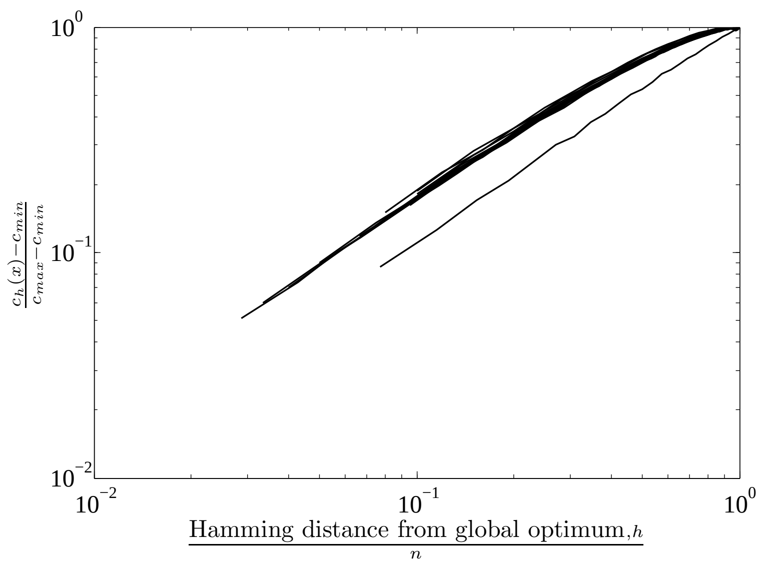


FIGURE 5.35: The normalised expected cost of configurations in Hamming sphere of radius h around a global optimum on a log-log scale for different size of the problem and different benchmarks.

Figure 5.35 shows the expected cost versus the Hamming distance from the global optimum on a log-log scale. Based on the scaling used in the figure, the data suggest that for all the random and benchmark problems we have studied, the expected cost grows polynomially with the Hamming distance. In one of our previous research on the landscape of the Travelling Salesman problem, we saw the expected cost grow linearly with the Hamming distance(see chapter 4). As the Travelling Salesman Problem is a particular form of the quadratic assignment problem, we can say that in general, the expected cost grows polynomially with the system size.

These curves show the existence of large-scale correlations in the fitness landscape. That is, the presence of a local optimum changes the expected fitness of configurations away from the mean fitness at all Hamming distances. Despite this large-scale structure, local search algorithms still fail to reliably find a global optimum. The reason for this is that the landscape is sufficiently rugged that it is not possible to exploit the long scale correlation using local fitness information alone.

In Figure 5.36, a density plot of the local optima as a function of their fitness and their Hamming distance from the best visited optimum is shown. A more detailed study on the density of local optima and their distances from the global optimum solution in the random and benchmark problems can be found in Merz and Freisleben (2000).

A landscape satisfies the big-valley hypothesis, if the closer a local optimum is to the global optimum, the fitter the optimum (Boese et al., 1994b). In the quadratic assignment problem we often have some low cost optima which are far from the global optimum. For example in Figure 5.36, there are many low cost local optima with the Hamming distance of 50 from the global optimum, which means that there could be low cost local optima at the farthest distance possible from the global optimum. A more detailed study on this property is found in Merz

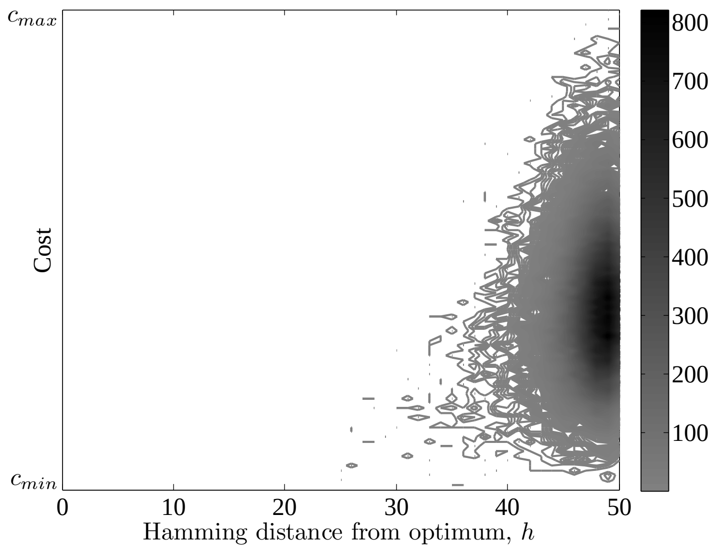


FIGURE 5.36: Density plot of local optima configurations as a function of their costs and Hamming distance from the best found optimum. The shading of the points shows the number of local optima at that cost and Hamming distance. The size of the problem is $n = 50$ and this is for a particular randomly drawn problem instance. The number of Hill-climbs is 10^6.

and Freisleben (2000). Furthermore, although there is a slight correlation between the fitness of a local optimum and its distance from a global optimum, there are plenty of fit local optima with a considerable distance from a global optimum, and many of unfit local optima close to a global optimum. Thus we cannot see the classic big-valley structure in the quadratic assignment problem instances. Nevertheless, the local optima close to any fit optimum, tends to be below the average cost.

Figures 5.37 and 5.38 show the average cost of the local optima versus the distance from the global optimum for different size of the problem for random and benchmark problems. The data for random problems are averaged over 100 randomly drawn problem instances and the number of descents is 10^5. The random problems show quite similar behaviour, while different benchmark problems show different behaviours. Apart from the slight differences, the data show an important property of the fitness landscape namely on average, the landscape has a valley shape. Note however that this analysis shows the average cost for configurations in a Hamming sphere h, around the global optimum and of course there are many local optima with different behaviour from the average. As the Hamming distance decreases, the average cost of the local optima from the global optimum increases, but for most of the benchmark problems and all the random problems, taking the last steps to Hamming distance of n, the cost slightly decreases. This is similar (and may be for the same reason) to the behaviour we saw in the cost of the configurations at a Hamming sphere around the local optima in MAX-SAT in chapter 2 and Graph-Colouring in chapter 3 landscapes.

Considering Figure 5.11 and Figure 5.37 together, the general shape of the landscape can be imagined. On average, the local optima closer to the global optimum have lower cost. On the

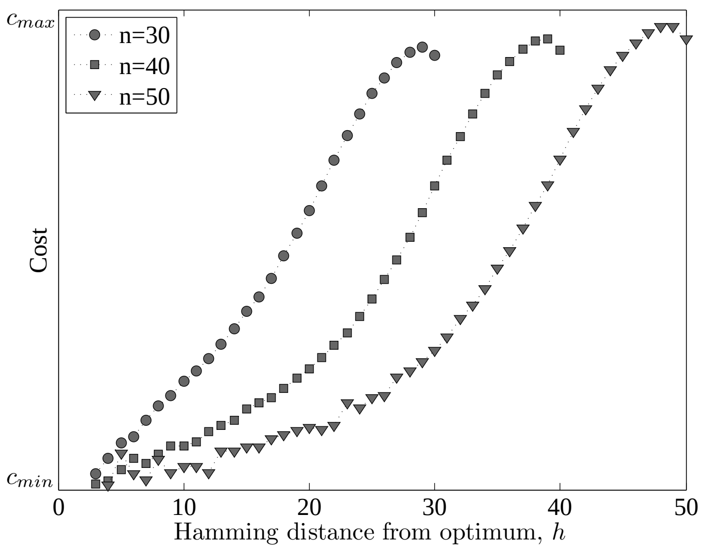


FIGURE 5.37: The average cost of the local optima versus the distance to the global optimum. This is for random problems and the data are averaged over 100 randomly drawn problem instances of different size. The number of descents is 10^5 on each problem instance.

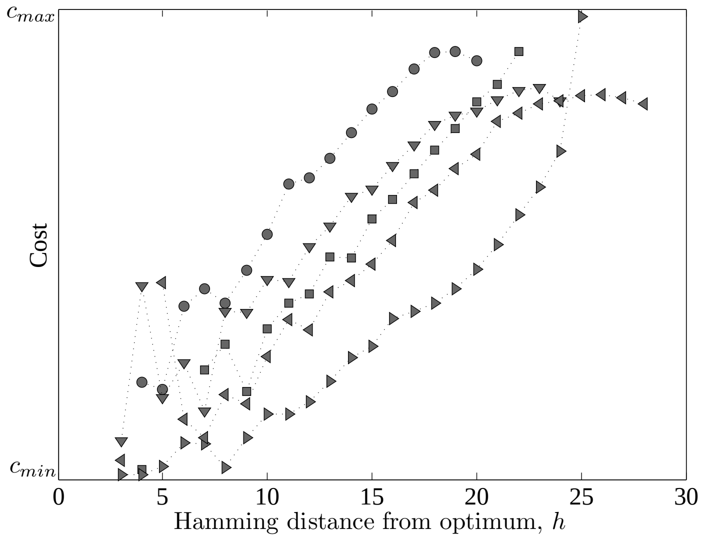


FIGURE 5.38: The average cost of the local optima versus the distance to the global optimum. This is for five benchmark problems of different size. The number of descents is 10^5 on each problem instance.

other hand each local optimum has a bowl shape basin of attraction. Therefore, the landscape is made of a valley consisting of these bowls, and on average, the closer the bowls are to the global optimum, the deeper (on average) they are.

In order to show how the curve representing the cost of the local optima in a Hamming sphere changes with the system size, Figure 5.39 exhibits the curves for different size of the problem. The data indicate that the valley (if we may call it) consists of two parts. At the first part, close to the global optimum, there is a plateau (note that this is not the same as the plateaux we see

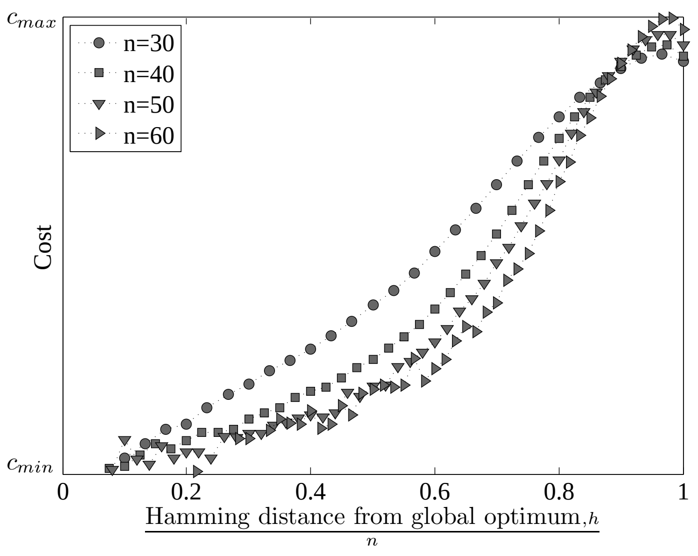


FIGURE 5.39: The average cost of the local optima versus the distance to the global optimum divided by n for different size of the problem. This is for random problems and the data are averaged over 100 randomly drawn problem instances of different size. The number of descents is 10^5 on each problem instance.

in Figure 5.13, here we are talking about local optima which may be far apart). Then as we get further from the global optimum, we see a slope. According to the data presented in Figure 5.39, we see that as n increases, the slope (fitness distance correlation) diminishes close to the global optimum. This would provide less heuristic information for a population based algorithm to locate the global optimum.

5.3.2 Basins of Attraction

The region in the search space which leads a local-search algorithm to a particular local optimum is usually called the "basin of attraction" of the local optimum. We can empirically measure the "basin of attraction" of a local optimum by repeatedly jumping to a certain Hamming distance from the local optimum, running the local-search algorithm until it reaches a local optimum and measuring the number of times it ends up at the same local optimum we had jumped from. This allows us to measure the return probability, starting from a configuration in a given Hamming sphere. Figure 5.40 shows the return probability for the global optimum and some inferior local optima for a particular quadratic assignment problem instance. The global optimum has been visited for 9.79% of the times, and the least visited one has been hit just once in 10^5 hill-climbs. We observe that the local-search algorithm has around 92% chance of reaching the global maximum, if it starts within a Hamming distance of 10. This chance for the least visited local optimum is 5%.

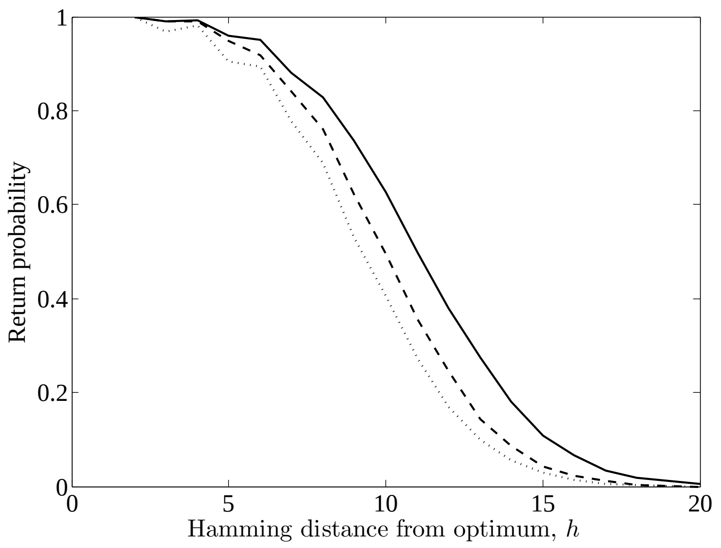


FIGURE 5.40: Return probability starting from a Hamming sphere of radius h versus h. This is for the global optima and some inferior local optima in a particular random problem with size of $n = 20$. The empirical return probability is computed by running 10^4 hill-climbs starting at randomly chosen configurations in each Hamming sphere.

5.4 Conclusion

We study different real-world and random problems of different size and show how the properties of the landscape of the problems change from one problem benchmark to another. We also show how the real-world problems are different from random problems.

The properties we study in this paper are as follows.

- The auto-correlation of the fitness landscape is studied as a measure for the landscape ruggedness. Our analysis shows that the correlation length depends on the ratio of the similar values in the problem matrix.
- One of the important properties of the fitness landscape is the time taken by a local search algorithm to get to the local optima. We study the time to local optima for different benchmark and random problems. The studies show that although there are some differences among the real-world problems, they show quite similar behaviours, both for the random and real world problems the time to local optima seems to grow polynomially with the system size as $T \approx n^{1.3}$. We also study the size of the plateaux and its relationship with the time to local optima and size of the system. We show that the random problems do not show the existence of plateaux in the landscape, while due to their structure, many of the real-world problems have plateaux, a property that makes the local search algorithms take longer to get to a local optimum. The analysis shows that there is a correlation between the size of the plateaux and the number of similar values in the problem matrix. The other property in the time to local optima is the number of cliffs the local search algorithm sees

during its search process. We showed how this property is related to the system size; it grows polynomially as $N_{cliffs} \approx n^{1.4}$. The quite similar growth rate of the number of cliffs and the time to local optima suggests that in the Quadratic Assignment problem, it is mainly the number of cliffs which determines how long it takes for the local search algorithms to find the local optima, than the size of the plateaux.

- The other property we study is the number of times a local optimum is visited as a function of its cost. The analysis shows that as the cost of the local optima increases, the probability of finding the local optimum decreases exponentially. This desirable property means that the local search algorithms have higher chance of finding the better local optima.

- The number of local optima seems to grow exponentially with the system size, although the rate seems to be greater in some benchmarks than the others. We show that there is an inverse relationship between the number of local optima and the time to local optima for problems with certain size. This means that for a particular system size, the larger the number of the local optima, the less time it takes for the local-search algorithms to find them. Therefore there exists a trade-off between the hardness in terms of the time to local optima and the hardness in terms of the number of local optima.

- Comparing the histogram of the local optima at each cost with the probability distribution of the finding the local optima at each cost, it is clear that there is a bias towards the better local optima. We show that this bias decreases linearly with the system size, a property which means that finding the better local optima becomes more unlikely as the system size grows.

- Although the better local optima have higher chance of being visited, the gap between the cost of the global optimum and the expected cost of the local optima found by the local-search algorithms increases with the system size. We show that the gap increases linearly with the system size. At the same time, the variance of the expected cost of the found local optima divided by the average cost of the local optima decreases. This means that as the problem size grows, the chance of finding the better local optima decreases.

- The probability of finding the globally optimum solution is studied and it is shown that as the system size grows, the probability of finding the global optimum shrinks exponentially. Our studies show that the real-world problems follow the same rule.

- One important property of the fitness landscape of the Quadratic Assignment problem is the correlation between the cost of the local optima and their distance to the global optimum. Although there is a slight correlation between the fitness of a local optimum and its distance from a global optimum, we showed that there are plenty of fit local optima with a considerable distance from a global optimum, and many unfit local optima close to a global optimum. Thus we cannot see the classic big-valley structure in the quadratic assignment problem instances. Although, the local optima close to any fit optimum, tends to be below the average cost.

Chapter 6

Comparison

In this chapter we perform a comparison between the four problems, namely The MAX-SAT, Graph-Colouring, Travelling Salesman and Quadratic Assignment problems studied in this research. By studying the landscape properties of these problems we try to point out the similarities and the differences between these problems. Many of the graphs in this section are the same graphs presented in the previous sections and are presented here for the sake of comparison.

6.1 Bulk Properties

In this section we study two properties that are not directly related to the local optima. The first one is the auto-correlation function and the second is the mean fitness in a Hamming-sphere about a particular configuration. These measures tell us about the average ruggedness of the landscape and long-range correlations in the landscape.

6.1.1 Auto-Correlation

The empirically measured log-auto-correlation for the MAX-SAT, Graph-Colouring, Travelling Salesman and Quadratic Assignment problems for different instance sizes is represented in Figure 6.1. In all cases the fall off of the logarithm is almost linear, indicating an exponential fall off with a characteristic "correlation length" (note that the deviation for large τ which is most prominent for TSP is most likely due to sampling errors). The correlation length is proportional to the system size for all the problems. We observe that in all cases the landscapes are fairly smooth, and become smoother as the system size increases.

For MAX-SAT we show the auto-correlation for different ratios of clauses to variables, α. For $\alpha = 2$ almost all instances have a relatively large fraction of satisfiable configurations. In contrast, at $\alpha = 10$, almost no instances are satisfiable, with instances around $\alpha = 4.3$ marking the phase-transition. The behaviour of the auto-correlation gives no indications of this rather

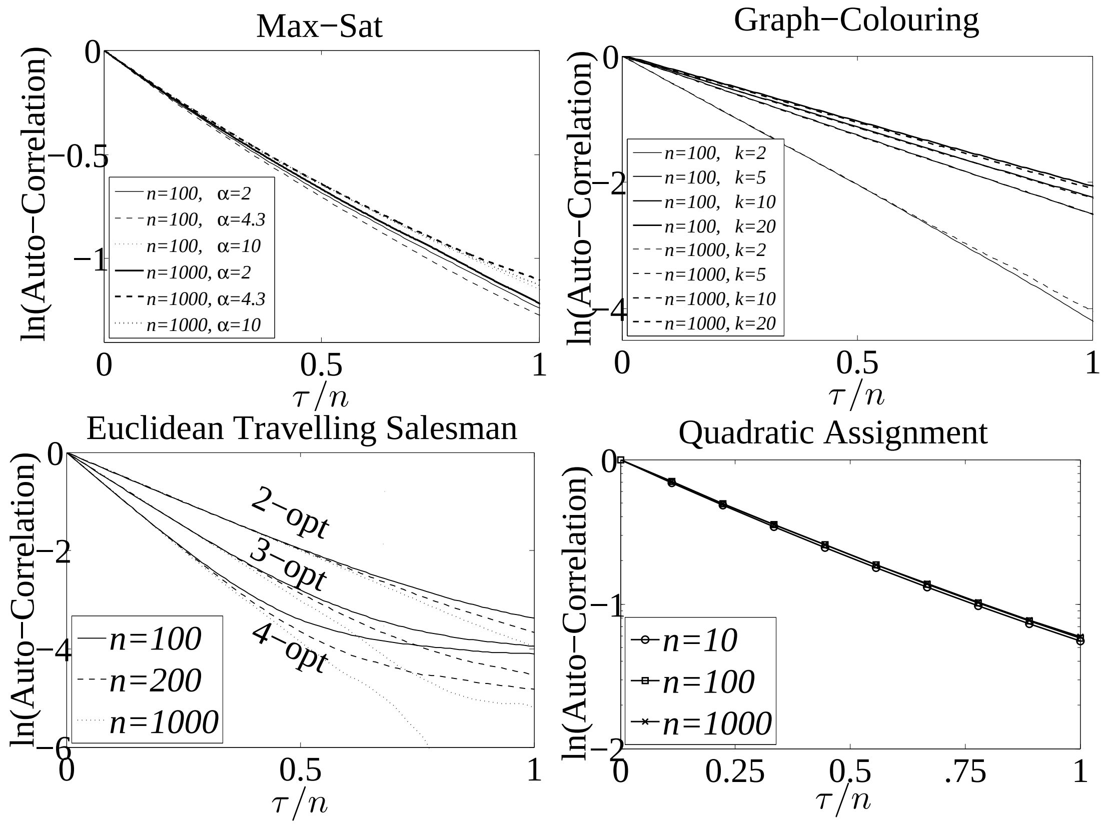


FIGURE 6.1: The logarithm of the auto-correlation of the fitness landscape for the MAX-SAT, the Graph-Colouring, the Travelling Salesman and the Quadratic Assignment problems for different values of n and parameters plotted against the time difference τ/n. The results are for randomly generated problem instances. The number of steps is 10^9.

important change in the behaviour of the problem. For Graph-Colouring the correlation length grows as $n(1-1/k)/2$ (see Stadler (1996); Tayarani-N and Prügel Bennett (2011)), where k is the number of colours. The phase-transition depends non-linearly on both k and n. As the difficulty of the problem instances varies depending on the proximity of the phase-transition, we observe that the auto-correlation function is not a good measure of problem difficulty. For the Euclidean-TSP we show the auto-correlation function for different search operators. As would be expected, the more disruptive the move operator the more quickly the correlation function falls.

6.1.2 Expected Cost in Hamming Sphere

The auto-correlation shows the existence of a strong correlation in the cost of neighbouring configurations. Another measure which shows the same behaviour is the expected cost of a configuration in a Hamming Sphere from some reference configuration. This can often be computed analytically (see chapter 3). We show plots of this, starting from different low-cost minima in Figure 6.2. The costs fall off from their initial value to the average cost as we move towards the expected Hamming distance between randomly chosen configurations. This behaviour arises because the cost functions involve a sum of terms each of which depends on only a few variables. Thus, perturbing a few variables results in a relatively small change in the cost of a solution. One consequence of this behaviour is that low cost solutions have a long range influence on the fitness landscape and that the lower cost minima tend to have larger basins of attraction than the higher cost minima.

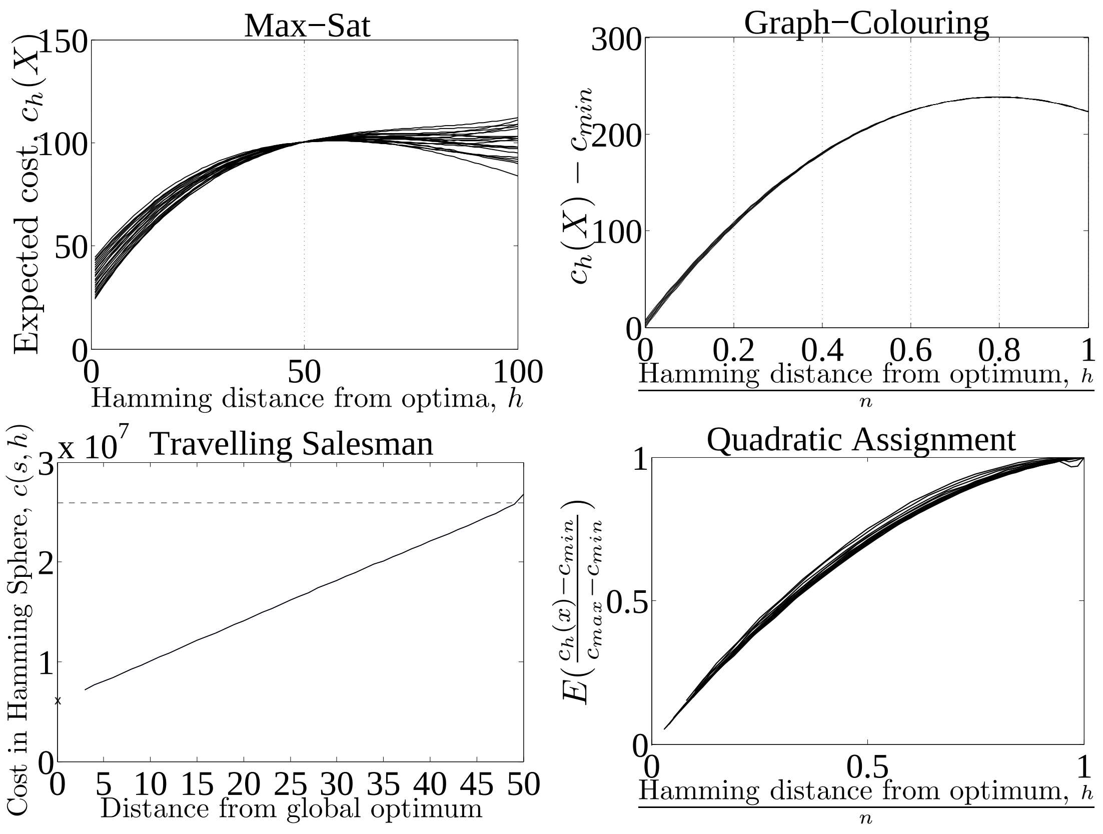


FIGURE 6.2: Expected cost of the configurations in a distance sphere for the most local optima at each cost for different problems of different size. For the MAX-SAT $\alpha = 8$ and for the Graph-Colouring $k = 5$. The data are averaged over 10^5 different runs.

6.2 Properties of Local Optima

In the rest of this chapter we concentrate on properties of local optima. These were found using the exhaustive local search, described in section 1.2, many thousands of times to obtain a representative sample of local optima with large basins of attraction. As we argue later we believe that this sample includes almost all low cost solutions and in particular the global optima.

6.2.1 Time to Local Optimum

We consider properties about the search process, starting with the time (number of moves) to reach an optimum. Note that we count the number of moves used, rather than the number of fitness evaluations, since in highly optimised local search algorithms data structures are used to avoid or speed up fitness evaluations. The time complexity of the local search is dominated in these algorithms by the update of the local search after making a move. We discuss this further at the end of this section. In this section, we distinguish between the constraint-satisfaction problems (MAX-SAT and Graph-Colouring) and the other two. As discussed earlier, the constraint satisfaction problems undergo a phase-transition from an easy phase to a hard phase, as the number of constraints increases (assuming the number of variables is kept fixed). In Figure 6.3 we show the empirically measured mean time to reach a local optimum versus α for MAX-SAT and versus k for Graph-Colouring.

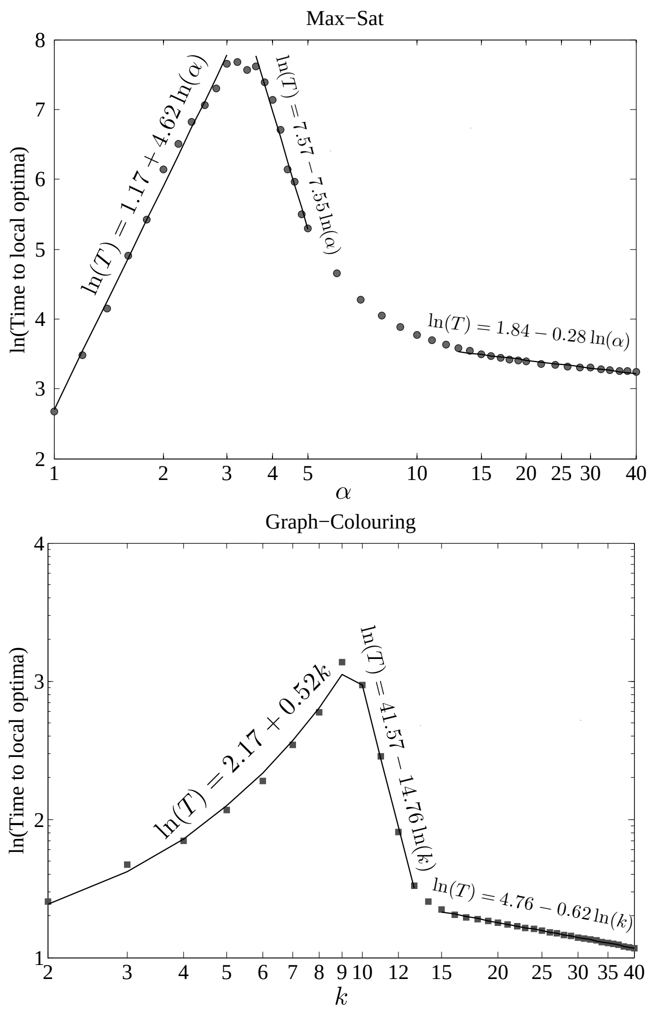


FIGURE 6.3: Logarithm of the time to reach a local optimum for the problems with phase transition. The data for the MAX-SAT problem are plotted versus α on a log-log scale, where the size of the problem is $n = 50$. The data for the Graph-Colouring problem are plotted against the number of colours k on a logarithmic scale for $n = 100$. Each data point represents the mean over 100 instances and 10^5 hill-climbs per instance.

For MAX-SAT, small α corresponds to the easy-phase with a small number of constraints. For small α there is a relatively high proportion of the configurations that satisfy all the constraints. A local search algorithm will consequently find a globally optimal solution quickly. However, as α increases towards the phase transition (which occurs around $\alpha = 4.3$), local search slows down dramatically as the searcher explores very large plateau regions. Above the phase-transition the search plateau regions become less common and the search rather rapidly finds a local optimum. Exactly, the same pattern of behaviour is observed for Graph-Colouring. Note that for graph colouring we have examined changing the number of colours k. A large number of colours makes it easier to find a satisfying solution; thus, the easy phase occurs on the right side of Figure 6.3(b) and the hard side on the left (which is a mirror image of what occurs in Figure 6.3(a) for MAX-SAT). The phase-transition occurs for this size of instance at $k = 9$.

In the MAX-SAT problem, graph representing time to local optima versus α consists of three stages. From $\alpha = 1$ up to $\alpha = 3.2$ (phase transition), on a log-log scale, the time to local optima versus α fits a straight line, suggesting that the time grows polynomially as $T \approx \alpha^{4.16}$. After the phase transition, there are the second and the third stages in which the time decreases polynomially. At the second stage, as α increases, the time decreases rapidly by $T \approx \alpha^{-7.55}$ (note that empirically it is difficult to distinguish between an exponential increase and a large polynomial—we fitted with a polynomial as it gives a slightly better fit). After reaching a certain point, the third stage begins, where the time decreases less rapidly as $T \approx \alpha^{-0.28}$.

A quite similar behaviour is seen in the Graph-Colouring problem. The time to reach a local optimum also consists of three stages. On a logarithmic scale, at the first stage, from $k = 2$ to $k = 9$ (the chromatic number), the data fit a straight line, meaning that for k up to the chromatic number, the number of steps grows exponentially with the number of colours. The second and the third stages of the graph show different behaviours. On a log-log scale, the data for these two stages fit a straight line, suggesting that the time to local optima decays polynomially with the number of colours. Note that this time consists of the time taken on the plateaux, plus the number of improvement moves the local search algorithm sees during its search process. We will study each component separately in the rest of this research.

The time to reach a local optimum as a function of the instance size is found to grow as a slow polynomial of the system size. This is shown in Figure 6.4. For TSP we find a similar behaviour in a suit of instances (see 4). The MAX-SAT and Graph-Colouring data are for instances in the unsatisfiable (hard) phase.

We observe that the number of moves to reach the local optima grows sub-quadratically in all cases. To compare the run times between the different problems we also need to take into account the time taken to make a move. This depends on the implementation of the local search operator. The best algorithm known by the authors for solving MAX-SAT can perform a move in $O(1)$ operations, while that for Graph-Colouring takes $O(n)$ (Prügel-Bennett and Tayarani-N., 2011). TSP can be much slower, as it has a large neighbourhood ($O(n^3)$ for 3-opt), and each move can require a non-local rearrangement of the tour. Efficient implementations of TSP usually restrict the neighbour and use sophisticated data structures to store the tour. The quadratic assignment problem is considerably simpler than TSP, but still has a large neighbourhood to explore ($n(n-1)/2$) compared to n for MAX-SAT and $(k-1)n$ for Graph-Colouring. Thus, the clock time can grow faster than Figure 6.4 suggests. Nevertheless, the time taken to reach a local optimum is not a limiting constraint in finding a global optimum.

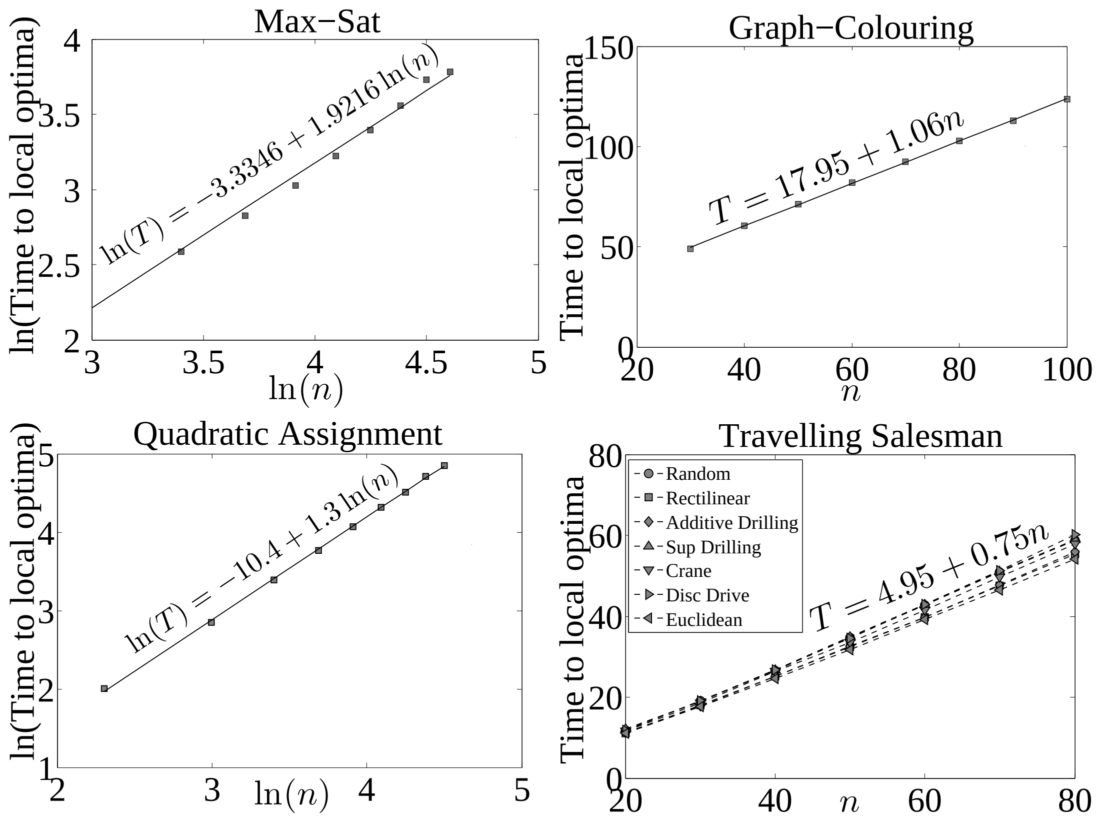


FIGURE 6.4: Time to reach a local optimum for the problems versus the system size. For the Graph-Colouring problem $k = 7$, and for the MAX-SAT problem $\alpha = 8$. Each data point represents the mean over 100 instances and 10^5 hill-climbs per instance.

The analyses provided for different combinatorial problems suggest that the time to local optima for these problems grows sub-quadratically with the system size. This, may seem to mean that in terms of the time to find a local optimum, the NP-Hard problems (or at least the ones studied in this research) become polynomially hard as the system size grows. However, this does not mean that finding the best solution to a NP-Hard problem can be performed in polynomial time. There are other properties related to the hardness of the problems. The number of local optima and the correlation between the size of the basin of attraction of a local optimum and its cost are other important properties which are targeted in the next sections.

The average time to reach local optima hides details of how the search algorithms make progress. To study this we examine the change in cost as a function of the number of steps averaged over many different runs. This is shown in Figure 6.5 as a density plot of the cost of a solution versus the number of steps (plotted on a logarithmic scale) for many different runs for the Graph-Colouring and the quadratic assignment problems. For Graph-Colouring (and similarly for MAX-SAT) there is a characteristic dog-leg shaped curve, with rapid initial progress until the knee of the dog's leg is reached. At this stage on most runs a local optimum has been reached and no further progress is made, but on a small proportion of runs, large plateaux are reached and the rate of progress slows down significantly once this happens. The probability of reaching a large plateau increases considerably in the proximity of the phase-transition. In contrast, for quadratic assignment (and similarly for TSP), this second phase is not observed. This is a consequence of the fact that for quadratic assignment (and TSP) configurations with the same fitness are rare, so there are no large plateaux.

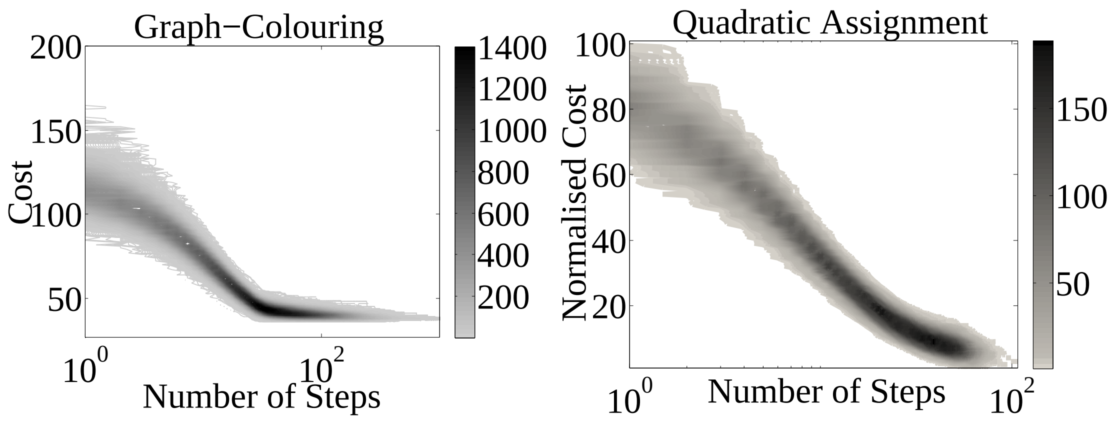


FIGURE 6.5: The density plot of the record of the steps taken by the local search algorithm to get to the local optima for 10^5 different search process, starting from random configurations. The cost of the configurations at each step is represented against number of steps on a logarithmic scale. The size of the problem is $n = 50$. For the Graph-Colouring problem the number of colours is $k = 5$.

Note in Figure 6.5 that we have plotted the number of steps on a logarithmic scale; thus, initially, the frequency of improving steps and the average size of an improving step is significantly larger than that which occurs later on in the search.

The time it takes for a local-search algorithm to find a local optimum consists of two components. One is the number of improving moves (cliffs) the search algorithm sees, and the other is the plateaux on which the algorithm wanders while searching for an improving move. These two properties change as the parameters of the problems change. Since there is no plateau in the landscape of the Quadratic Assignment and the Travelling Salesman problems, the time to local optima is entirely made of improving moves. The landscape of Graph-Colouring and MAX-SAT however, consists of both improving moves (when the search has reached a cliff edge) and plateaux. For the Graph-Colouring problem, we saw that up to the chromatic number, the time to local optima grows exponentially with the number of colours. Figure 6.6 shows the contribution of number of improving moves and the number of steps on plateaux in the time to local optima. Up to the chromatic number, the number of improving moves needed to reach a local optimum grows with the number of colours. After reaching its peak at the chromatic number, the number of improving moves decreases with the number of colours. The number of improving moves also depends on the system size. It seems that it grows linearly with the system size. On the other hand the number of steps on the plateaux has a more important role in the time to local optima. Quite similar to the behaviour we saw in time to local optima in Figure 6.3, the graph representing the size of the plateaux in the landscape consists of three stages. At the first stage, from $k = 2$ up to the chromatic number, the size of the plateaux grows exponentially with the number of colours. At the second stage, the size decreases polynomially with the number of colours until it reaches zero. And finally is the third stage at which the size of the plateaux is zero. This means that at the third stage (far above the chromatic number), the search algorithm directly reaches the local optima without wandering on the plateaux and the time to local optima consists entirely of the improving moves.

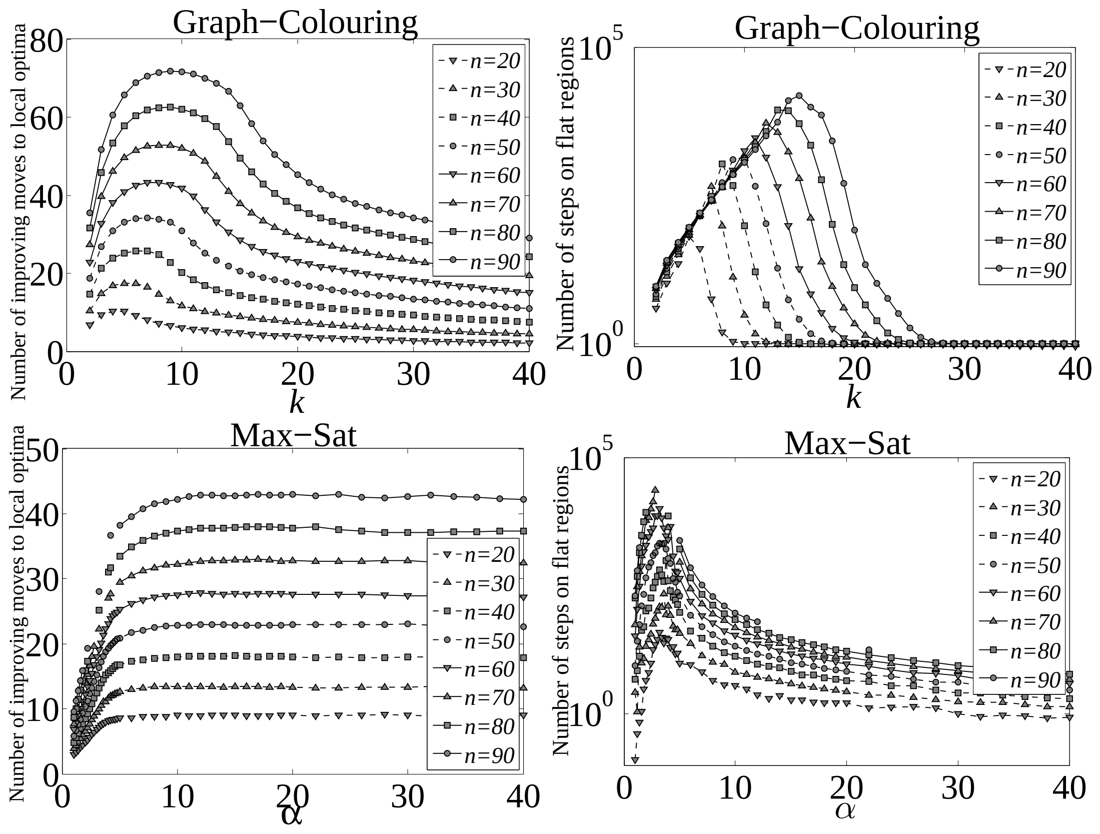


FIGURE 6.6: Number of improvement moves and the size of the plateaux the local search algorithm sees during its search process versus the constraint parameter for different size of the problems. This is for the Graph-Colouring and the MAX-SAT problems. Each data point represents the mean over 100 instances and 10^5 hill-climbs per instance. The data for the number of steps on the plateaux are plotted on a logarithmic scale.

A rather similar behaviour is seen in the MAX-SAT problem. Up to the phase-transition, the number of improving moves need to reach a local optimum grows with α. Here, in contrast to the Graph-Colouring problem, after the phase-transition, the number of improving moves remains constant as α grows. Figure 6.6 also shows the number of steps on the plateaux in the MAX-SAT problem. Quite similar to the Graph-Colouring problem, the size of the plateaux in the MAX-SAT problem has three stages. From $\alpha = 1$ up to the phase-transition, the size of the plateaux grows rapidly with α. After the phase-transition, the size decreases polynomially with α, but unlike the Graph-Colouring does not reach zero.

The analyses provided here show that unlike the Quadratic Assignment and the Travelling Salesman problems, which show no plateau in their landscape, for the problems with highly degenerative objective function, the size of the plateaux has a much more important role in the time to local optima than the number of improving moves has. This means that for these problems, most of the time of the local search algorithms is spent wandering on the plateaux (note that the data for the number of steps on the plateaux are plotted on a logarithmic scale). When wandering on the plateaux, it speeds up search if we do not check repetitive solutions. This property suggests that the Tabu search might be a more appropriate algorithm for the degenerative objective functions, particularly around their phase transition.

6.2.2 Local Optima Properties

In this section, we explore properties of the local optima. As discussed before, these were found by performing multiple runs of the exhaustive search algorithm. The mean number of visits for local optima binned according to their costs is shown in Figure 6.7. For all four problems shown, the probability of visiting a local optimum appears to fall off exponentially with the cost. Similar behaviour was found for instances at other parameter values and also for different types of TSP (see chapter 4). The only exception was for TSP with random "distances", where there was little, if any, correlation between the cost of the local optima and the probability of visiting them. We attribute this to the lack of significant long range structure in this problem.

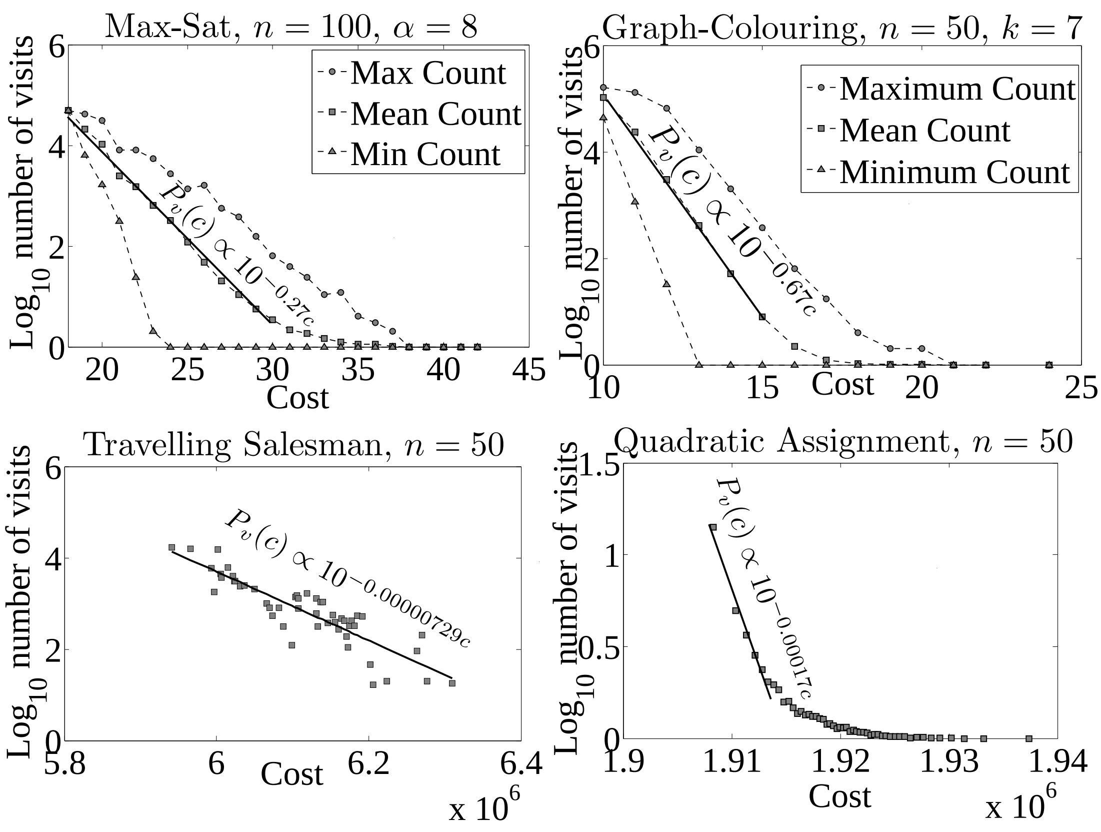


FIGURE 6.7: Number of local optima are hit at each cost for different problems. For the Graph-Colouring and MAX-SAT problems we show the maximum, mean and minimum number of times the local optima are hit at each cost level. For the Quadratic Assignment and the Travelling Salesman problems it is highly unlikely to have more than one local optimum at each cost, so the data show the number of times each local optimum is hit versus its cost. The size of the problem is $n = 100$ for the MAX-SAT problem and $n = 50$ for the other problems. For the MAX-SAT problem $\alpha = 8$ and for the Graph-Colouring $k = 7$. The number of descents is 10^6.

In Figure 6.7, we see that the mean probability of visiting a local optimum at cost c is approximately exponential $P_v(c) \propto 10^{-\lambda c}$. The gradient, λ, depends on the properties of the instances. In Figure 6.8 we show this gradient as a function of the system size and of α for MAX-SAT and k for Graph-Colouring. For MAX-SAT this gradient is, to a first approximation, independent of the problem size, but quite dependent on the ratio of clauses to variables, α. The situation is more complicated in Graph-Colouring, although as the position of the phase-transition does not scale linearly with n, this is not surprising. For both problems, as the number of unsatisfiable constraints increases, there is a small bias towards the low cost solutions.

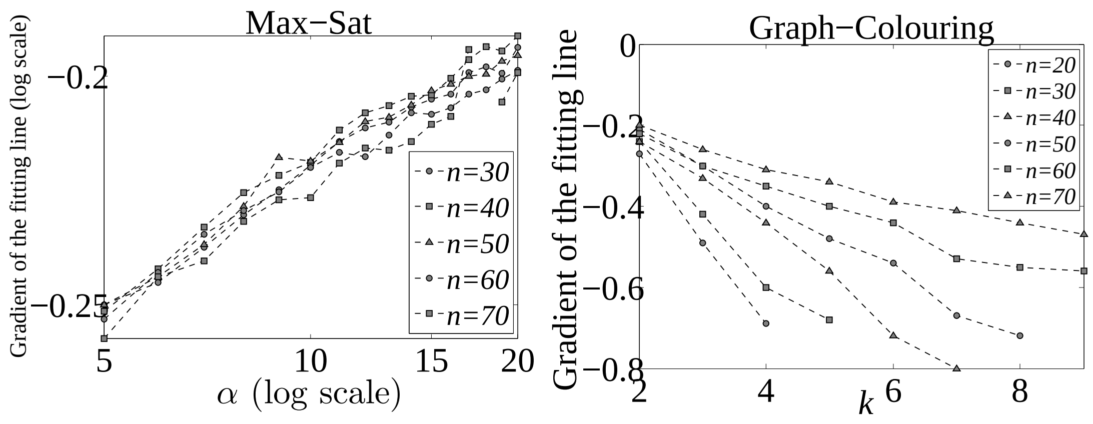


FIGURE 6.8: The decay rate of the mean number of time the local optima at each cost level are hit (the gradient of the fitting line to the data in Figure 6.7) for different size of the problems versus the constraint parameters for the MAX-SAT and the Graph-Colouring problems. The data are averaged over 50 randomly generated problem instances and 10^6 runs on each. The data for the MAX-SAT problem are represented on a log-log scale.

Better local optima have, on average, exponentially greater chance of being found than less fit local optima, which clearly helps in finding a global optimum. However, there are exponentially more local optima at lower costs. We illustrate this in Figure 6.9 where we show the proportion of local optima in different cost bins, and the probability of reaching a local optimum in a particular cost bin. We see clearly that the expected cost of a local optimum found by local search is substantially less than the mean cost of all local optima.

Figure 6.9 suggests that there is a fairly high probability of finding a global optimum. This is true for moderate size problems of the type investigated here. However, the number of local optima grows rapidly with the instance size. We will see later that this results in the probability of local search finding a global optimum decreasing exponentially. Figure 6.10 shows the scaling behaviour of the number of local optima at different costs. We show the logarithm of the number of local optima in each cost bin, divided by some power of n chosen so that the peak of the curves are roughly similar. For quadratic assignment the vast majority of local optima were only seen once. It was thus not possible to estimate the scaling behaviour for the number of local optima using this method.

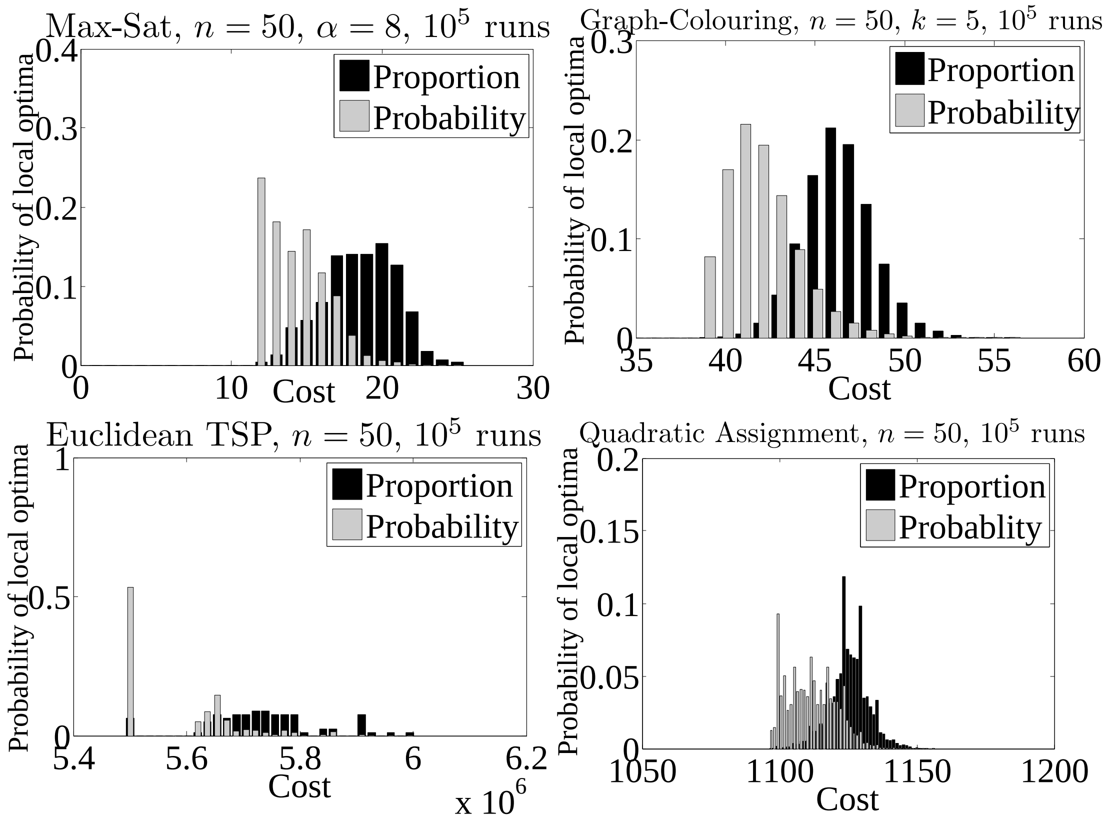


FIGURE 6.9: Histograms showing the proportion of local optima at each cost value and the probability of finding a local optimum of each cost for one randomly generated instance of different problems. The size of the problem is $n = 50$ for all the problems. The constraint parameters are set to $k = 5$ and $\alpha = 8$.

The curves indicate that the number of local optima at each cost grows roughly as

$$N(c) \approx e^{n^a g(c/c_a v)},$$

where $g(\cdot)$ is some approximately parabolic function (Graph-Colouring is the slight outlier as it is difficult to find a scaling relation due to the fact that the chromatic number does not scale linearly with n). Each problem has a very different scaling behaviour with n. This is seen even within problem classes. Different types of TSP have different scaling behaviour. Note that for Graph-Colouring we count local optima that differ by a permutation symmetry as the same local optimum. Thus with the Hamming neighbourhood, due to the permutation symmetry, there are in fact $k!$ more local optima. To estimate the growth in the number of local optima for quadratic assignment we measured the proportion of randomly sampled configurations which were local optima. This was only practical for problems up to size 12. From these very small instance sizes we estimated that the number of local optima grows approximately as $\exp(0.63n - 2.2)$.

The number of local optima also depends on the other details of the instances. In Figure 6.11, we show the total number of local optima for MAX-SAT versus α and n, and for Graph-Colouring versus k and n. Note that because of the simple scaling behaviour of MAX-SAT, we have scaled the y axis to remove the main variation in n. We observe that the number of local optima grows

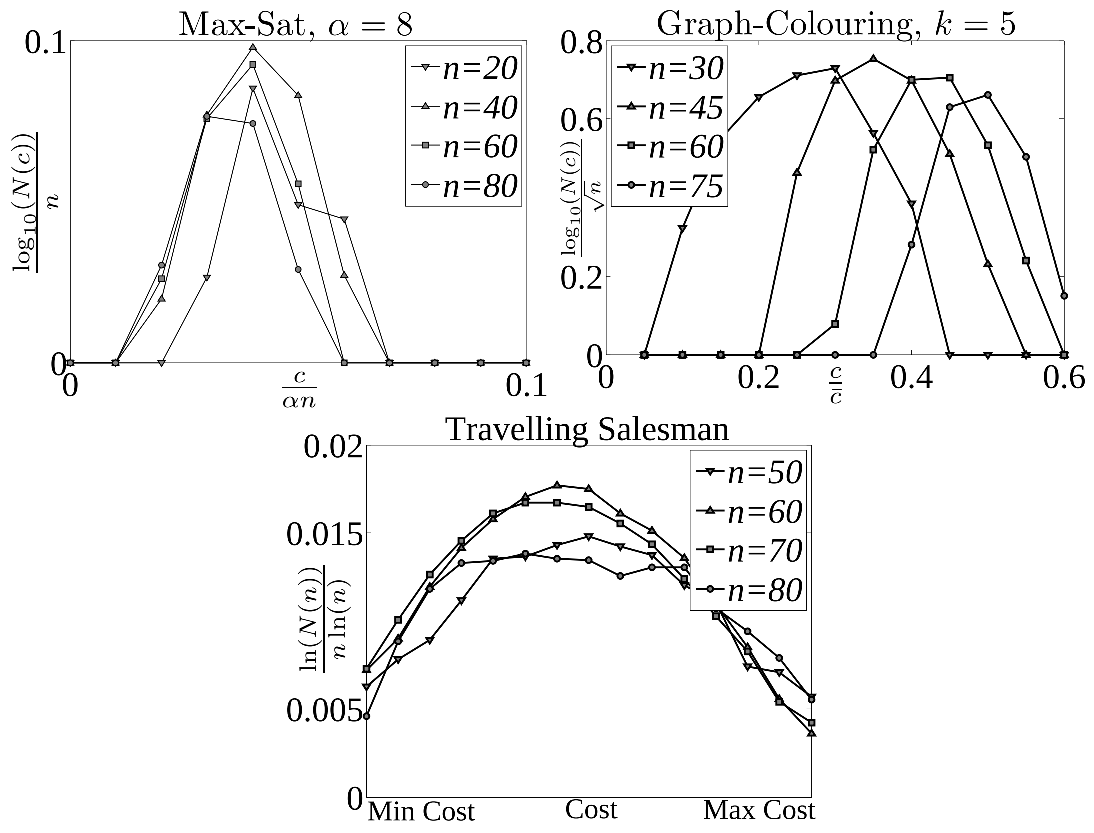


FIGURE 6.10: Number of local optima at each cost scaled by the system size, n versus the cost divided by the average cost for different problems. This is averaged over 100 randomly generated problem instances for 10^5 runs.

linearly (and rather slowly) as we increase the number of constraints αn. The scaling behaviour of Graph-Colouring is more complicated and we just show the raw number of local optima as a function of n and k (note that for large n and k we will be substantially under sampling the number of local optima, so the rate of growth may be significantly greater than shown). The behaviour is more complicated than that for MAX-SAT, since increasing k both increases the size of the search space (which grows as k^n), and makes it easier to satisfy the constraints. These two effects tend to balance, resulting in the number of local optima remaining fairly constant for a considerable proportion of the graph.

6.2.3 Global Optima

As the instance size increases, the gap between the cost of typical configurations found by local search and the global minimum cost grows. This is shown in the top two graphs in Figure 6.12 for MAX-SAT and Graph-Colouring. The y-axis has been scaled by n and the average cost (n^2) respectively, so these gaps look linear. By plotting the MAX-SAT result versus $1/n$, it is possible to extrapolate back for large n. The typical gap in this limit is around $0.03n$. The extrapolation

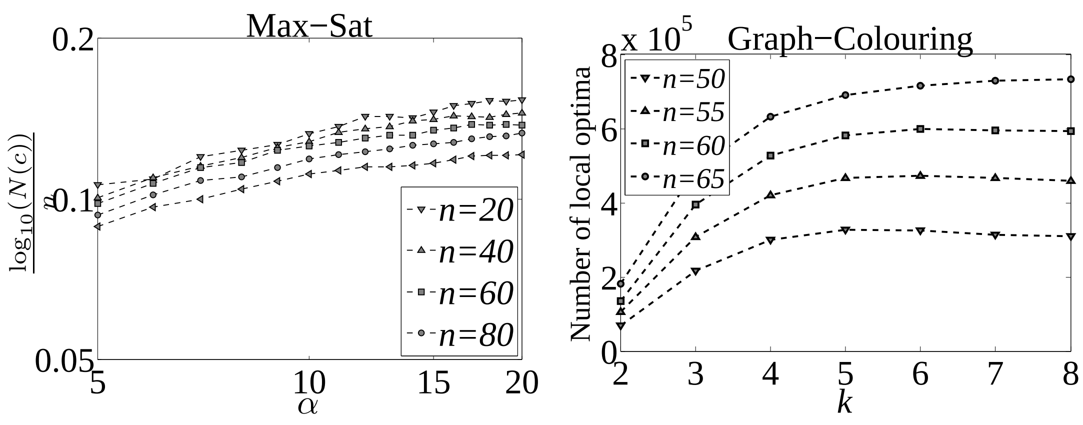


FIGURE 6.11: The number of local optima versus the number of colours k and α for the Graph-Colouring and the MAX-SAT problem. The data are averaged over 100 randomly generated problem instances of different size. The number of runs is 10^6. The data for the MAX-SAT are plotted on a log-log scale.

for Graph-Colouring is more complicated. A plausible fit to the data available would be

$$c_{min} = \bar{c}\left(1 - \frac{0.22}{\sqrt{n}} + O\left(\frac{1}{n}\right)\right),$$

where $\bar{c} \approx n^2/(2k)$ is the average cost. What is clear is that the gap between the minimum cost and the average cost found by a local search is of order n^2.

The bottom graphs in Figure 6.12 show the variance in c/n for MAX-SAT and c/n^2 for Graph-Colouring. We see that this variance decreases towards zero in both cases, indicating that as n increases, it becomes less likely for local search to find local minima with a cost close to the global minimum cost.

In Figure 6.13, we plot the log-probability of local search finding a global minimum as a function of the instance size. The straight line fits are consistent with an exponential fall off in the probability of finding a local optimum for all these problems. Note that we have chosen parameters so that both MAX-SAT and Graph-Colouring are in their unsatisfiable (hard) phase. The exponents indicate the rate at which the problems become difficult (at least for the local search we have used). By this measure the quadratic assignment problem becomes harder, substantially faster than the other problems, followed by Graph-Colouring.

For Euclidean TSP and quadratic assignment the probability of two configurations having the same cost is very low and consequently there will nearly always be a unique global optimum. This is not the case for MAX-SAT and Graph-Colouring where there is only a relatively small number of cost levels. In Figure 6.14 we show a histogram of the number of local optima. In just over one third of cases the global optimum is unique, but in a reasonable number of cases there can be quite a few global optima.

Often the global optima are quite close to one another in Hamming distance, but on occasion uncorrelated global optima can exist. This is shown in Figure 6.15 where we have calculated

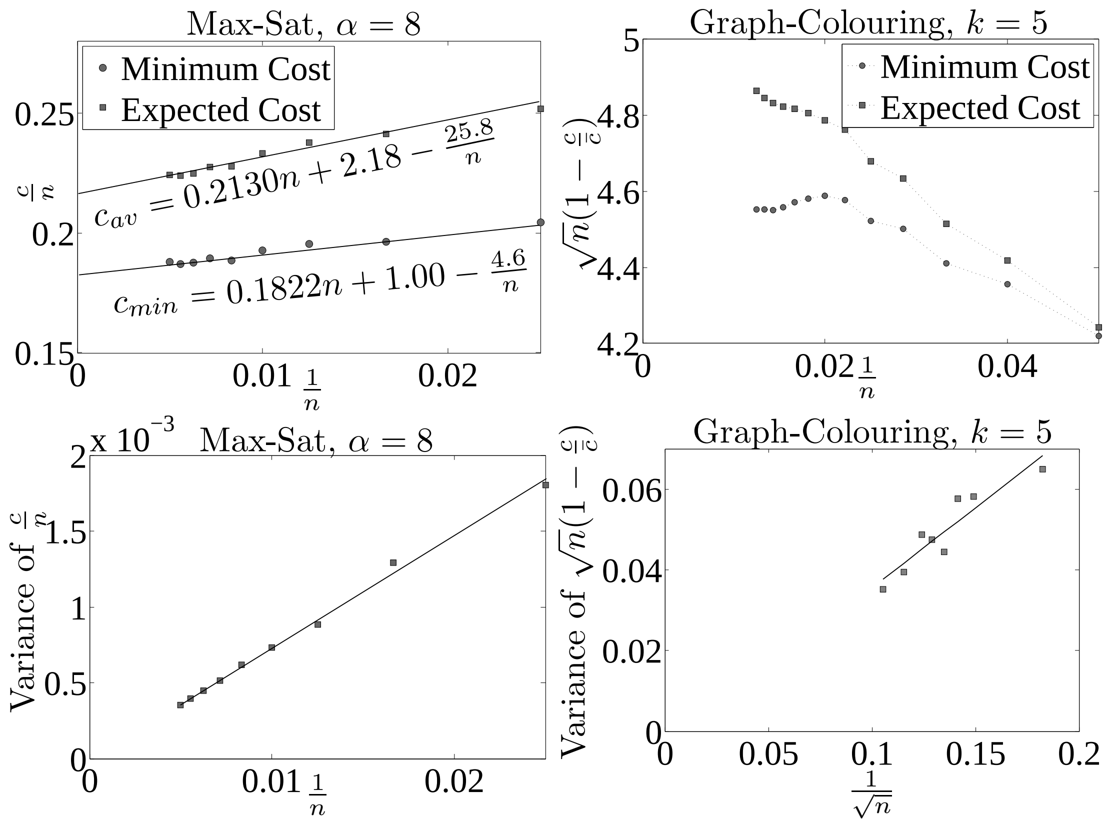


FIGURE 6.12: Plot of the expected maximum fitness (expected minimum cost for Graph-Colouring), the average fitness (average cost for Graph-Colouring) found by local search algorithm and the variance in the fitness of the local optima versus $1/n$ ($1/\sqrt{n}$ for Graph-Colouring). This is calculated by performing 10^5 hill-climbs on 100 randomly generated instances of the problem.

the Hamming distance between global optima averaged over a large number of instances with multiple global optima. The mean distance between randomly chosen pairs of configurations for MAX-SAT with $n = 100$ is 50. For Graph-Colouring we show both the Hamming distance and the partition distance between global optima (the partition distance being the minimum Hamming distance between two configurations up to a permutation in the colours). The expected Hamming distance between random pairs of configurations for this Graph-Colouring problem instance is 40, while the mean partition distance is around 33. In both problems we see that there can exist global optima which are unrelated to one another.

6.2.4 Proximity to Global Optima

Given that it is exponentially unlikely to find a global optimum, an important question is whether it is possible to infer information about the position of the global optimum (or a high quality optimum) from information provided by local optima. To study this, in Figure 6.16, we show a density plot of the number of local optima as a function of the cost and the distance from the nearest global optimum. For Graph-Colouring we use the partition distance, since otherwise the colour-permutation symmetry would hide the fitness distance correlation. For TSP we show

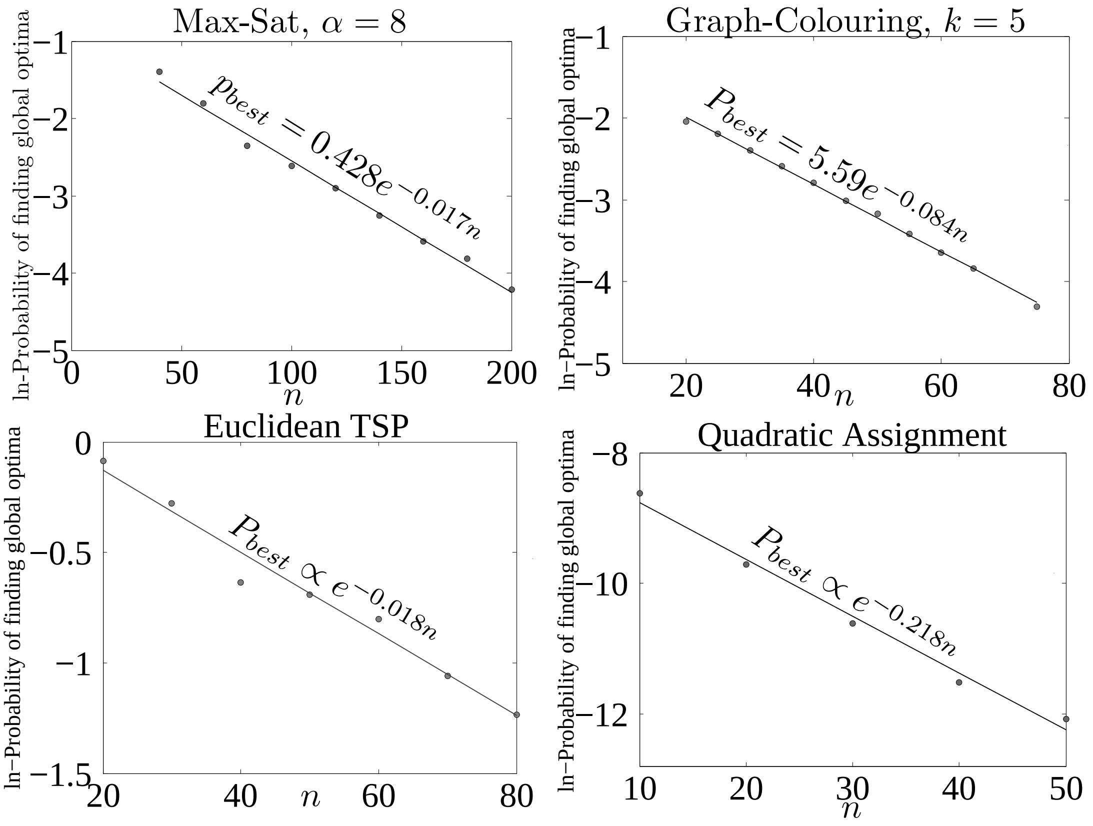


FIGURE 6.13: Natural log-probability of finding the maximum fitness optima versus n averaged over 100 randomly generated instances. The number of runs is 10^6. For the Graph-Colouring problem the number of colours is $k = 5$ and for the MAX-SAT problem $\alpha = 8$.

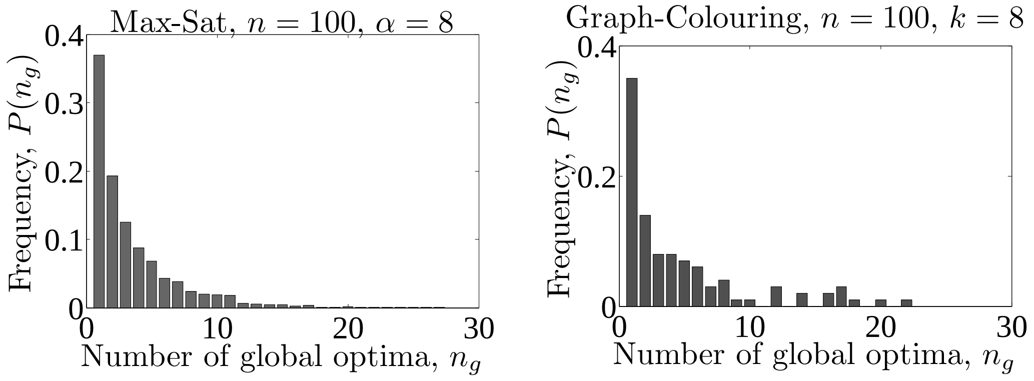


FIGURE 6.14: Histogram of the number of global optima for 10 000 random MAX-SAT and Graph-Colouring instances with $n = 100$ where $\alpha = 8$ and $k = 8$. The number of runs is 10^5 on each problem instance.

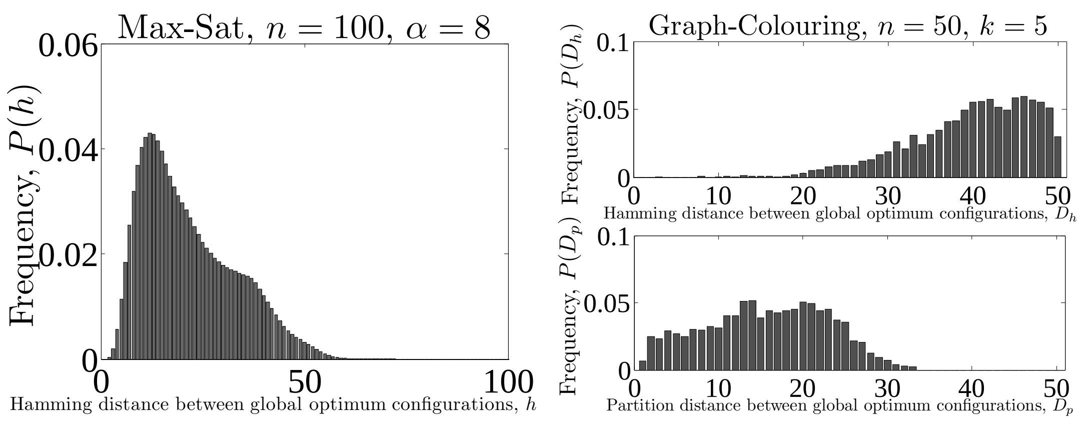


FIGURE 6.15: Histogram of Hamming distances between the global optima in the MAX-SAT and Graph-Colouring problems. For the Graph-Colouring problem we have also included the histogram of partition distances. The histograms are made from 10 000 randomly generated problem instances. For the MAX-SAT problem $n = 100$ and $\alpha = 8$, for the Graph-Colouring $n = 50$ and the number of colours is $k = 5$.

this both for Euclidean instances and for random instances. In all cases, there is some fitness distance correlation, although this is rather minimal for quadratic assignment and for TSP with random instances.

As fitter members of the population tend to be closer to a global optimum, this could allow population-based algorithms to search the landscape more effectively than local search. In MAX-SAT an effective search method is to perform a number of independent searches and then to move to the centroid of the best solutions found by local search (Qasem and Prügel-Bennett, 2010; Prügel-Bennett and Tayarani-N., 2011) (that is, the solution with the lowest mean distance to the best solutions). Although the centroid is usually at a higher cost than any of the solutions found by local search, when the search is restarted, the cost rapidly decreases and typically beats any of the solutions found by local search. We can understand this behaviour from Figure 6.16. Local search finds optima which tend to be at lower costs than the average optima. These are slightly correlated with the global optima. The centroid, although not itself an optimum, is correlated with the global optimum. Although this method does not typically find the global optimum solution, it will often find a solution close to a global optimum with a low cost.

In Graph-Colouring, one of the most effective algorithms over the last 10 years for solving dense random graphs is a hybrid genetic algorithm (Galinier and Hao, 1999; Tayarani-N and Prügel Bennett, 2011). Again we can understand this in terms of the density plot shown in Figure 6.16. This algorithm uses a specially tailored crossover which finds a child with a small partition distance between its two parents. The crossover is hybridised with a local search. The local search finds solutions which are below average cost and thus tend to be correlated with a global optimal solution. Crossover produces new configurations which are no longer optimal, but tend to be more correlated with a global optimum. Reapplying local search tends to find a nearby above-average solution, which is likely to be even more correlated with the global solution.

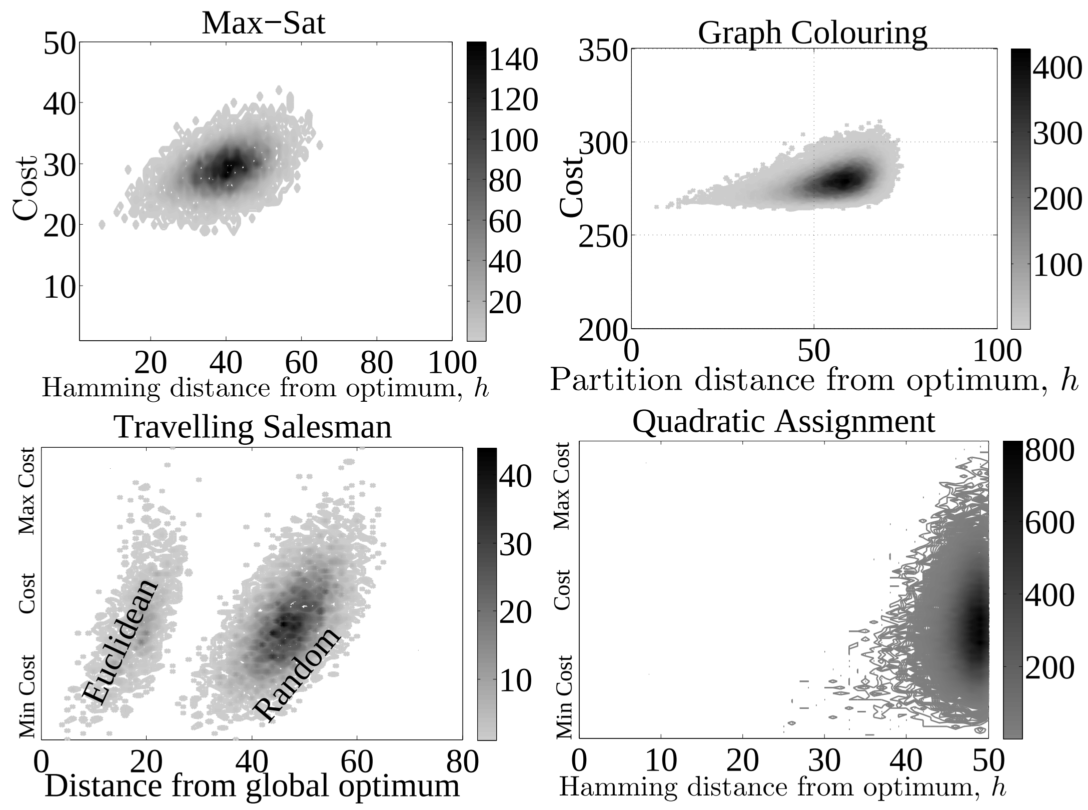


FIGURE 6.16: Density plot of local optimum configurations as a function of their cost and Hamming distance from the most frequently visited global optimum. For the MAX-SAT and the Quadratic Assignment problems we use the Hamming distance; for the Graph-Colouring problem the partition distance and for the Travelling Salesman problem the number of non-common edges are used as the distance measure. The number of colours in the Graph-Colouring problem is $k = 5$ and in MAX-SAT $\alpha = 8$. For the Travelling Salesman problem two types of the Random and the Euclidean are used. The number of runs is 10^6.

However, the fitness distance correlation is rather weak. In Figure 6.17 we show the average distance to the global optima for configurations in different cost bins. The axes have been rescaled to make the data fall on top of each other for different problem sizes (we were unable to do this for quadratic assignment).

For both MAX-SAT and Graph-Colouring we find that the mean distance to a global optimum appears to scale approximately as

$$d_{nearest} \approx n \left(\frac{c - c_{min}}{n} \right)^{0.33} .$$

Therefore the mean distance to the global optimum for a local optimum with cost of one greater than the global optimum scales as $n^{0.67}$. The number of configurations in a Hamming-ball of size h is given by $\binom{n}{h} \geq (n/h)^h$ so that the number of configurations in a Hamming ball with a radius of $d_{nearest}$ is greater than $n^{0.33n^{0.66}}$. Within this radius there are no lower cost configurations providing heuristic information about where to search. The situation for TSP is no better, with very fit solutions having in expectation around $0.2n$ edges different from the global solution. The number of configurations in a Hamming ball of this size is

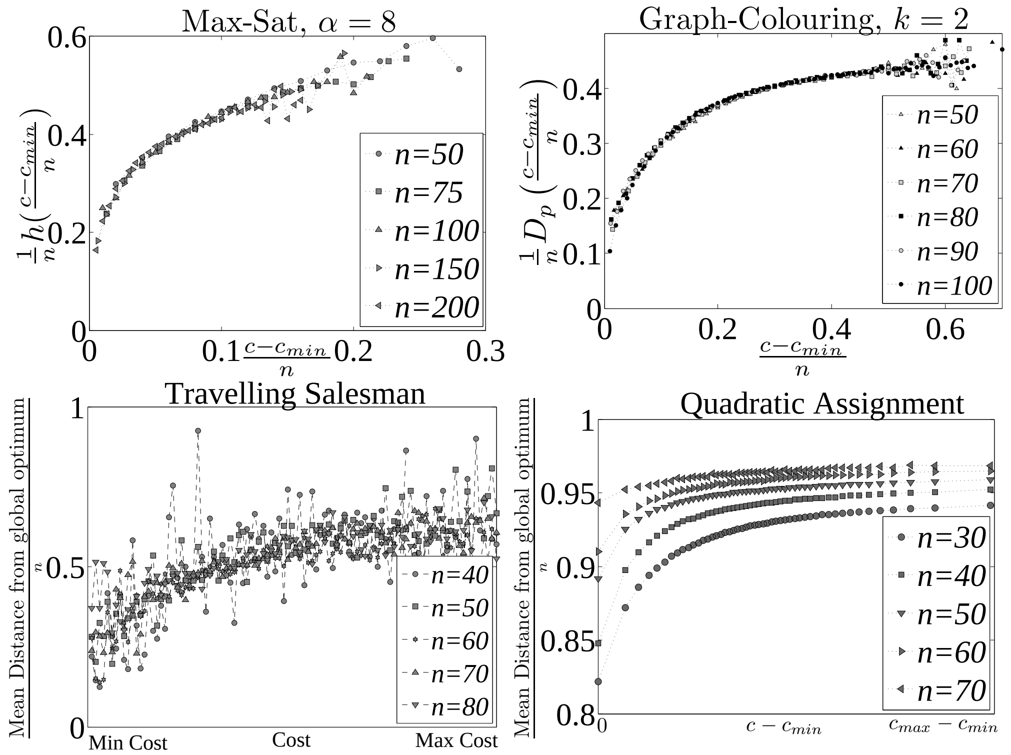


FIGURE 6.17: Measure of the mean distance to the closest global optimum from a local optimum of cost c for different problems of different size. The data are averaged over 100 problem instances and the number of runs is 10^6. For the MAX-SAT problem $\alpha = 8$, and for the Graph-Colouring the number of colours is $k = 2$.

$\binom{n}{0.2n}(0.2n)! = n!/(0.8n)! \approx n^{0.2n}$ (although not all these configurations are valid tours). The situation is even worse for quadratic assignment where the gap between a good solution and the global optimal solution is greater than $0.8n$ and the number of configurations in a Hamming ball of this size is $\binom{n}{0.8n}(0.8n)! = n!/(0.2n)! \approx n^{0.8n}$. Thus, finding a global optimum, even when starting from the next fittest local optimum, is still extremely challenging.

6.2.5 Principal Component Analysis

The local optima are correlated; however, a single measure of the correlation provides very little information about the directions in which the local optima differ. The very large number of these local optima may actually just vary in a relatively small number of ways. For example, in TSP we can imagine a situation where there are many small local alterations that are independent of each other. This can give rise to an exponential number of local optima by choosing different combinations of local variations. To explore the variation in local optima, we perform principal component analysis (PCA) on the set of local optima. For MAX-SAT we can compute PCA directly on the binary vectors representing the solutions. For Graph-Colouring we consider vectors where each component represents a pair of edges in the graph. We assign 1 to the component if the edges are connected to nodes of the same colour and zero otherwise. Clearly, this representation is invariant to colour permutations. For TSP we again consider a representation in terms of the occupancy of each possible edge in the tour. Finally, for quadratic assignment we consider the occupancy of an edge in the bipartite graph between facilities and locations.

This approach to exploring the set of local optima was introduced in chapter 4. That section shows that each eigenvector can be viewed as showing possible alternatives between different ways of solving TSP. It was also shown that, by projecting the data in the space spanned by the first few principal components, it was often possible to find clustering in the data that was not transparent otherwise. Often the high quality optima were confined to a single cluster, suggesting that this approach could potentially be used for guiding search.

In this research we consider only the spectrum of eigenvalues, that is the values of eigenvalues. Note that the expected reconstruction error of a local minimum is proportional to the sum of the eigenvalues that are dropped. Thus this spectrum provides a measure of the number of relevant directions in the space of local optima. In Figure 6.18 we show the eigenvalue spectrum for the four problems of interest.

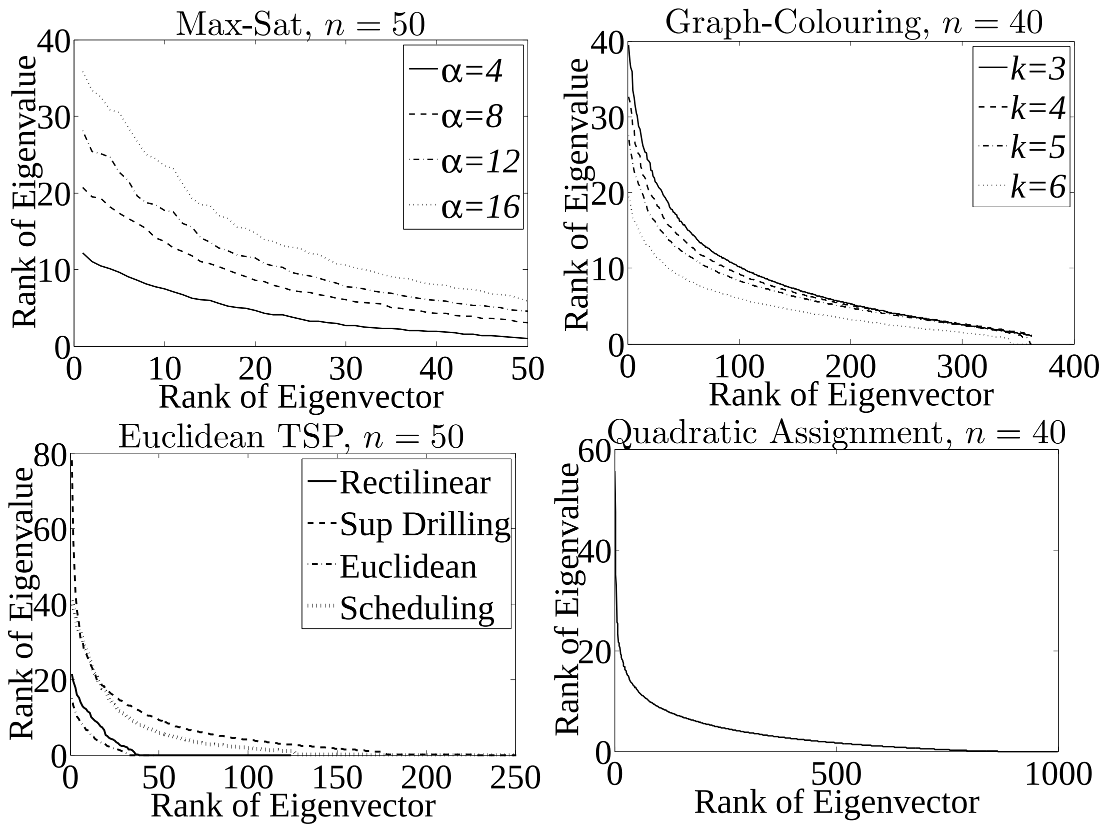


FIGURE 6.18: Plot of the eigenvalues of the covariance matrix of local optima versus their rank.

Note that the size of the eigenvalues measures the number of components in which the solutions differ from each other. A rapid fall off shows that the variation is dominated by only a few major directions, while a slower fall off shows that local optima can be constructed in very different ways. MAX-SAT has by far the smallest number of directions in which the data can differ. This reflects the much smaller search space. Note that in MAX-SAT the variation in the local optima increases considerably with α. This shows that increasing α considerably increases the size of the space in which the local optima sit. All the other problems have a very rapid initial drop off in the eigenvalue spectrum, indicating that there is a small number of directions where there are very prominent changes in the solutions. In Graph-Colouring we see that increasing k decreases the size of the space spanned by the local optima, despite increasing the size of the search space. In TSP we see that different problem types lead to very different eigenvalue structures. Euclidean TSP is somewhat atypical in being particularly easy to solve. This is reflected in the eigenvalue spectrum falling off faster than with other problems. Finally, we note that quadratic assignment has a large number of non-zero eigenvalues, which may in part explain its difficulty (although note that Graph-Colouring with $k = 3$ shows slightly more variation).

Chapter 7

Conclusion

This research provides a comparison between four very different combinatorial optimisation problems. We have had to be selective in the data we present. The first observation is the remarkable degree of similarity we observe in the four problems. Some of this is qualitative only, and appears more similar than it really is, due to selection of the scale of the axes. Nevertheless, there are substantial similarities. As we argued earlier, we believe the cause of these similarities is that the structure of the cost function is similar in so far as they are all sums of a large number of largely independent components (clearly they are not completely independent as this is what makes the problems hard). Furthermore, changing a small number of variables only affects a small number of components, so there is a long range correlation between costs. This results in a multi-modal fitness landscape with some correlation between local minima.

At least for the properties we examined (which we believe are all relevant to designing or choosing a search algorithm), this similarity seems to be much more important than the differences in the problem. These differences include the neighbourhood structure, the size of the search space and symmetries. In designing algorithms, these features are essential at an operator level (i.e., designing mutation or crossover operators for EAs). However, at a high level (e.g. using a population to explore the search space over long distances), these differences seem much less important and the variation within a problem class can be as significant as variations between problem classes. Being able to separate the low-level (short-scale) properties from the large-scale properties is clearly of the utmost importance in developing meta-heuristics.

Despite the similarities, there are some important differences between problems. Most significantly, in the problems we have studied, is the division between the constraint optimisation problems with a satisfiability phase transition and TSP and quadratic assignment which do not show the phase transition. Although all four problems behave similarly, when MAX-SAT and Graph-Colouring are above the phase transition, there is a significant qualitative difference around and below the phase transition. With a small number of constraints (i.e., significantly below the phase transition) MAX-SAT and Graph-Colouring have a significant proportion of configurations which satisfy all the constraints, and they tend to be easy to solve. Around the phase transition, there can be exponentially large low-cost plateaux which can dominate the search time. At the same time, some of these correspond to local rather than global optima. Efficient methods for navigating through this space, such as Tabu search or using additional heuristic information, can substantially speed up search for these problems.

None of the behaviour we observe is that surprising, although there are properties that would be difficult to guess and occasionally contradict commonly held believes. We outline some of these below.

- Auto-correlation is often used as a measure of problem difficulty, and yet, as we have seen, all four hard problems we have studied have a low correlation length. Furthermore the correlation length is oblivious to properties such as the phase transition where the problem difficulty changes drastically. In our opinion the importance of auto-correlation is often over played in studies of fitness landscapes.
- In all four problems the presence of a local optimum strongly influences the expected cost of a configuration up to a Hamming distance equal to the random Hamming distance between configurations. One consequence of this is that fitter local optima tend to have significantly larger basins of attraction than less fit local optima as their whole neighbourhood is on average fitter than that of a less fit local optimum.
- The time for local search to reach a local optima is poorly studied, although clearly of the utmost importance in designing heuristic search algorithms. The number of moves taken to reach an optimum grows sub-quadratically as a function of the problem size for all four problems investigated. However, it can grow exponentially large as we approach the phase transition (for those problems with a phase transition). The cause of this is the large amount of time spent on very large plateaux of the search space. The position of the phase transition is the most important determiner of the behaviour of the problem for constraint satisfaction problems. In the phase where it is not possible to satisfy many of the constraints, the constraint satisfaction problems (MAX-SAT and graph colouring) exhibit similar behaviour to TSP and quadratic assignment.
- In all problems investigated, the number of local optima grows exponentially as a small polynomial of the instance size. This measure in combination with the basins of attraction determines the difficulty of finding a global optimum.
- The probability of finding a global optimum falls off exponentially with the instance size; however, this fall off is often sufficiently slow that medium sized problems (often up to several hundred elements) can be solved quite efficiently by running local search multiple times. Interestingly, these sized problems are often found in benchmarking libraries. Extreme caution is required in extrapolating results obtained on such problems as repeated local search becomes ineffective when considering larger problems.
- In all problems very fit local optima often are a considerable distance apart. This is particularly evident in MAX-SAT and graph colouring where frequently there are multiple

global optima. These will often have small correlations with each other showing that it is often possible to obtain optimal solutions in very different ways.

- Despite the above observation there is a small but significant positive correlation between fitness and the proximity to a global optimum. The structure of this, however, can vary considerably between problem types. An evolutionary algorithm using selection and crossover (possibly hybridised with a local search algorithm) may be able to exploit this correlation.
- In all problems the mean distance from a global optimum to the next best optimum is sufficiently large that finding the global optimum by exhaustively searching the proximity of the next best optima is intractable in problems of any size.
- Finally, although the number of local optima increases exponentially in some small polynomial of the system size, there are often only a relatively small number of ways in which the local optima vary. This is revealed by the rapid fall off in the size of the principal components.

Optimisation is not easy. There are many properties that can affect the choice of search algorithm to be used. We explored a number of them in this research. However, there is a surprising, but pleasing uniformity in the behaviour of a lot of apparently different optimisation problems. This holds out the promise that the task of understanding fitness landscapes, for at least a large group of real world problems, is not endless. We believe there is the hope of modelling many of the properties of a fitness landscape for many different problems using only a small number of parameters; hence, providing a powerful classification system for landscapes. We have not attempted that in this research, but rather, provided the raw data, as we believe it may be of use to other researchers interested in understanding the anatomy of fitness landscapes.

Appendix A

Implementing GSAT

We can implement GSAT so that each step takes on average $\Theta(\beta^2\alpha)$ updates. To do so, we must maintain a count of the number of satisfied variables in each clause and the change in fitness caused by flipping each variable. A variable occurs on average in $\beta\,\alpha$ clauses. When a variable is flipped we have to change the count of the number of satisfied variables in each clause. We also have to change the cost for flipping variables for all the variables that occur in the clauses that have been changed by the variable flip. Since there are $\beta - 1$ variables in a clause, other than the variable being flipped, we have potentially to change the flip cost of, on average, $(\beta - 1)\beta\,\alpha$ variables. It is a tedious but straightforward exercise to perform the book-keeping described above.

A less trivial part of the implementation is to be able to choose which variable to flip given a list of flip costs. Since there are n variables, searching this list would dominate the run time. There are two types of variables that need to be maintained, improving moves and neutral moves. Typically, the total number of improving moves throughout a run will be less than $2^{-\beta}\alpha n$. Since the run time to reach a local optimum grows super-linearly, the vast majority of moves will not involve a fitness improvement. Therefore, the improving moves do not have to be implemented efficiently. The more challenging issue is to maintain a set of fitness-neutral moves. In particular we want to be able to add elements, delete elements and choose a random element in constant time. Binary trees or hash tables do not allow this; however, there is a simple, although not well known, data structure that does this. We call this a bounded set.

The bounded set implementation works provided that 1) the number of objects that could be put into the set are both known when the set is created, and 2) they can be stored in memory. In our case, our list of neutral moves can only involve the n variables—thus this easily fits in memory. We implement the bounded set using two arrays. The first, *index*, array is a fixed-size array of size n which is used to point to elements in the second array. The second, *member*, array contains the elements in the set. We also keep a counter of the number of elements in the set. If the elements are not in the set, we assign a value of -1 to those elements in the index array. In Figure A.1 we show a set of up to 10 possible elements containing elements 5, 7, 2 and 8.

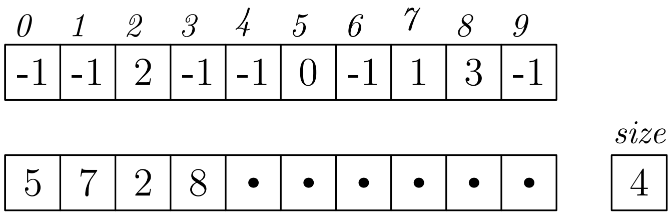


FIGURE A.1: Illustration of the two arrays used to implement a bounded set containing elements 5, 7, 2 and 8. The top, index, array holds the indices of elements in the bottom, member, array.

To add an element, say 3, to the set we put it into the next available position (position 4) in the member array and update the element at position 3 in the index array to 4. We finally increment the set size. To remove an element, 5 say (assuming the array is as shown in Figure A.1), we check its position using the index array and find it is in position 0. We then move the last element, 8, in the member array into this position (updating the position of element 8 in the index array), decrement the element count, and set the 5^{th} element in the index array to -1. To find a random element we just select a random number from 0 to the size of the set and choose the element in that position in the member array.

WALKSAT can be run as efficiently as GSAT. The only modification needed is to keep a bounded set of the unsatisfied clauses.

Appendix B

Estimating the Number of Optima

In estimating the number of local optima, we treat each set of optima at a given fitness separately. To simplify the notation we drop the subscripts indicating the fitness, for example, we denote the parameters of the Gamma distribution by a and b rather than a_f and b_f. Recall from section 2.3.4 that we shall assume that the probability of visiting an optimum, i, is gamma distributed, and the likelihood of visiting the optimum in N trails is Poisson distributed. To compute the likelihood of visiting a local optimum, at fitness f, n_i times we integrate over the unknown probability, Z_i, of visiting the local optimum

$$\begin{aligned}\mathbb{P}(n_i|a,b) &= \int_0^\infty \mathbb{P}(n_i|z_i)\,\gamma(z_i|a,b)\,\mathrm{d}z_i \\ &= \frac{b^a}{(b+N)^{a+n_i}}\frac{N^{n_i}}{n_i!}\frac{\Gamma(a+n_i)}{\Gamma(a)}.\end{aligned}$$

Given a list of the number of times each local optimum at fitness f is visited $(n_i|f_i = f)$, the log-likelihood is given by

$$\begin{aligned}\mathscr{L} &= \log\left(\prod_{n_i|f_i=f}\mathbb{P}((n_i)|a,b)\right) \\ &= n^t\big(\bar{n}\log(N) + a\log(b) - (a+\bar{n})\log(b+N) \\ &\quad - \log(\Gamma(a))\big) + \sum_i\left(\Gamma(a+n_i) - \log(n_i!)\right)\end{aligned}$$

where n^t is the total number of local optima at fitness f, and $\bar{n} = \sum_i n_i/n^t$ is the mean number of times a local optimum at this fitness is visited. To maximise the likelihood we set the derivative of $\mathscr{L}$ with respect to a and b to zero. After rearranging we find

$$a = \frac{\bar{n}}{\varepsilon\psi(a) - \frac{1}{n^t}\sum_i\psi(a+n_i) - 1}, \qquad b = \frac{Na}{\bar{n}}$$

where $\psi(x)$ is the digamma function which is defined as $\psi(x) = \mathrm{d}\log(\Gamma(x))/\mathrm{d}x$. Notice that we have a self-consistency equation for a, which, can be easily solved by bisection. The function

$$f(a) = \frac{\bar{n}}{\varepsilon\psi(a) - \frac{1}{n^t}\sum_i \psi(a+n_i) - 1}$$

has the property $f(0) = 0$ and $f'(0) = 0$, while for large a

$$f(a) \sim a + \frac{\sigma^2 - \bar{n}}{2\bar{n}} \pm O\left(\frac{1}{a}\right)$$

where $\sigma = \frac{1}{n_t}\sum_i n_i^2 - \bar{n}^2$. Thus, there must be a solution for finite $a > 0$ provided $\sigma^2 > \bar{n}$ since the function $f(a)$ starts out smaller than a for some sufficiently small a, but finishes larger than a. For $\sigma^2 \leq \bar{n}$, the maximum-likelihood solution occurs when $a \to \infty$ which corresponds to $\gamma(x_i|a,b)$ being a delta function at $x_i = \bar{n}$. This agrees with our intuition that, in this case, we can explain all the fluctuations in the number of times an optimum is visited through the Poisson process, and thus the maximum-likelihood distribution occurs when all the optima have the same probabilities of being found.

To obtain an accurate model of the distribution of probabilities of finding an optimum, we should also include the optima that we have not observed. Of course, we do not know this. However, we can estimate this number for a given set of parameters, a, b and n^t The expectation of not finding a local optimum is given by $\left(\frac{b}{b+N}\right)^a$. The expected number, n^u, of local optima that are not observed are thus given by

$$n^u = n^t \left(\frac{b}{b+N}\right)^a = n^t \left(\frac{a}{a+\bar{n}}\right)^a$$

We have $n^t = n^u + n^o$ where n^o are the number of observed local optima. We can find n^u by starting from $n^t = n^o$ and computing the expectation of n^u. This is then used to update n^t. Eventually this converges although the convergence is rather slow and benefits from using Aitken's δ^2-process (Press et al., 2007, Section 5.3). Empirically it is found that if the probability of being visited in N trials falls below 1 then the likelihood is optimised by $a = 0$ and $n^u = \infty$. This is clearly a quirk of the probabilistic model we are using and reflects the fact that there is not enough data to make a reasonable fit. If we just extrapolate the expected probability of visiting a local maximum, we see from Figure 2.10 that to visit a local maximum at a fitness of 757 at least once would require around 10^{11} samples which is computationally infeasible. Thus, obtaining a reliable estimate of the number of local optima at low fitness is extremely challenging.

Appendix C

Fitness Variance in Hamming Sphere for MAX-SAT

Computing the variance in the fitness of the configurations in a Hamming sphere depends on the structure of the clauses, and, in particular, which pairs of clauses share common variables. An exact computation of the variance, though possible becomes quite complicated. However, for randomly drawn instances where each clause is independent, we can obtain a reasonable approximation by assuming the probability of two clauses being satisfied is equal to the product of either being satisfied. This is true only in expectation for random configurations; however, it provides a reasonable estimate to the variance found in typical instances.

The variance is given by

$$\sigma^2 = \mathbb{E}_h\Big(s_0^2(\boldsymbol{x}')\Big) - \mathbb{E}_h\Big(s_0(\boldsymbol{x}')\Big)^2$$

where we have used the shorthand

$$\begin{aligned}\mathbb{E}_h\Big(q(\boldsymbol{x}')\Big) &= \mathbb{E}\left(q(\boldsymbol{x}')|d(\boldsymbol{x}',\boldsymbol{x}) = h\right)\\ &= \frac{1}{\binom{n}{h}} \sum_{\boldsymbol{x}'\in\mathscr{X}_h(\boldsymbol{x})} q(\boldsymbol{x}')\end{aligned}$$

for an arbitrary function q. Now

$$\begin{aligned}s_0^2(\boldsymbol{x}') &= \left(\sum_{i=1}^{m} [\![g_i(\boldsymbol{x}) \in \mathscr{S}_0]\!]\right)^2\\ &= \sum_{i=1}^{m} [\![g_i(\boldsymbol{x}) \in \mathscr{S}_0]\!]\\ &\qquad + \sum_{i\neq j=1}^{m} [\![g_i(\boldsymbol{x}) \in \mathscr{S}_0]\!]\,[\![g_j(\boldsymbol{x}) \in \mathscr{S}_0]\!]\end{aligned}$$

where we use the fact that since an indicator function equals either 0 or 1, the square of the indicator function is equal to itself. Thus $\sigma^2 = \sigma_1^2 + \sigma_2^2$, where

$$\sigma_1^2 = \sum_{i=1}^{m} \left(\mathbb{E}_h\Big([\![g_i(\boldsymbol{x}') \in \mathscr{S}_0]\!] \Big) - \mathbb{E}_h\Big([\![g_i(\boldsymbol{x}') \in \mathscr{S}_0]\!] \Big)^2 \right)$$

$$\sigma_2^2 = \sum_{i \neq j=1}^{m} \Bigg(\mathbb{E}\left([\![g_i(\boldsymbol{x}') \in \mathscr{S}_0]\!] \, [\![g_j(\boldsymbol{x}') \in \mathscr{S}_0]\!] \right) - \mathbb{E}\left([\![g_i(\boldsymbol{x}') \in \mathscr{S}_0]\!] \right) \mathbb{E}_h\Big([\![g_j(\boldsymbol{x}') \in \mathscr{S}_0]\!] \Big) \Bigg).$$

(Note σ_1^2 and σ_2^2 are not themselves variances—σ_2^2 can and often is negative). As we showed in section 2.4.1

$$\mathbb{E}_h\Big([\![g_i(\boldsymbol{x}') \in \mathscr{S}_0]\!] \Big) = \sum_{l=0}^{\beta} p_l \, [\![g_i(\boldsymbol{x}) \in \mathscr{S}_l]\!]$$

where $p_l = \binom{n-\beta}{h-l} / \binom{n}{h}$. Since any clause is in just one equivalence class $\mathscr{S}_l$ then

$$\mathbb{E}_h\Big([\![g_i(\boldsymbol{x}') \in \mathscr{S}_0]\!] \Big)^2 = \sum_{l=0}^{\beta} p_l^2 \, [\![g_i(\boldsymbol{x}) \in \mathscr{S}_l]\!] .$$

Thus σ_1^2 is equal to

$$\sigma_1^2 = \sum_{l=0}^{\beta} (p_l - p_l^2) s_l(\boldsymbol{x}).$$

Since clauses are drawn independently, we would expect, for a random configuration, $\boldsymbol{x}$, that the probability that two clauses, i and j, are both satisfied is in expectation

$$\mathbb{E}_h\Big([\![g_i(\boldsymbol{x}) \in \mathscr{S}_0]\!] \, [\![g_j(\boldsymbol{x}) \in \mathscr{S}_0]\!] \Big) = \mathbb{E}_h\Big([\![g_i(\boldsymbol{x}) \in \mathscr{S}_0]\!] \Big) \mathbb{E}_h\Big([\![g_j(\boldsymbol{x}) \in \mathscr{S}_0]\!] \Big)$$

so that σ_2^2 cancel. This is only true when averaged over all possible clauses—it is not true for any particular pair of clauses. However, due to the large number of pairs of clauses it provides a acceptable approximation for typical randomly drawn instances. However, the configurations $\boldsymbol{x}$ we consider are not randomly drawn configurations, but are fit configurations so that the argument that they are independent of the clauses is flawed. This leads to a small but systematic discrepancy in the prediction of the variance. In Figure C.1, we compare the empirically measured variance in the fitnesses of configurations in a Hamming sphere of radius h from a global optimum configuration and the approximation assuming $\sigma_2^2 = 0$.

Although there is clearly a systematic error with the theoretical approximation, it translate into an almost negligible error when we consider the standard deviation. We illustrate this in Figure C.2, where we show both the mean fitness and the mean fitness ± 1 standard deviation

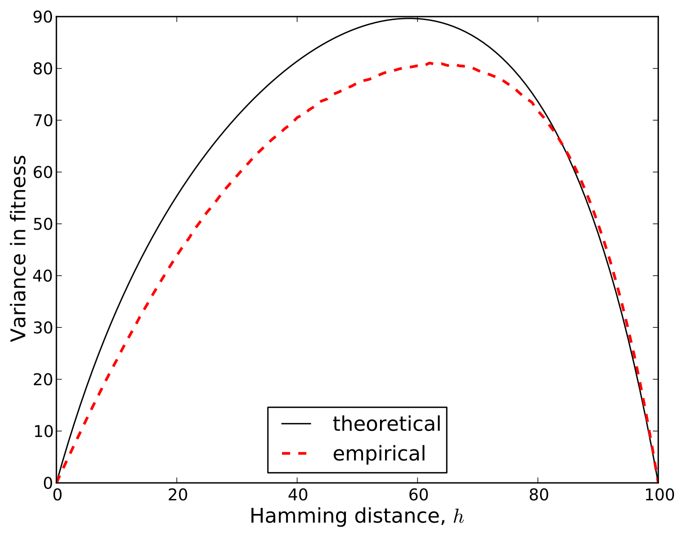


FIGURE C.1: Comparison of the variance in a Hamming sphere of radius h from a global optimum configuration measured empirically and using the approximation $\sigma_2^2 = 0$.

computed using the formula above and measured empirically. As we observe ,the approximation (i.e., assuming $\sigma_2^2 = 0$) provides a perfectly adequate estimate of the size of the fluctuations. In Figure 2.27, where we show the fluctuations for an instance of size $n = 10\,000$, the discrepancy between the empirically measured fluctuations and the theoretical approximation are smaller than the thickness of the line.

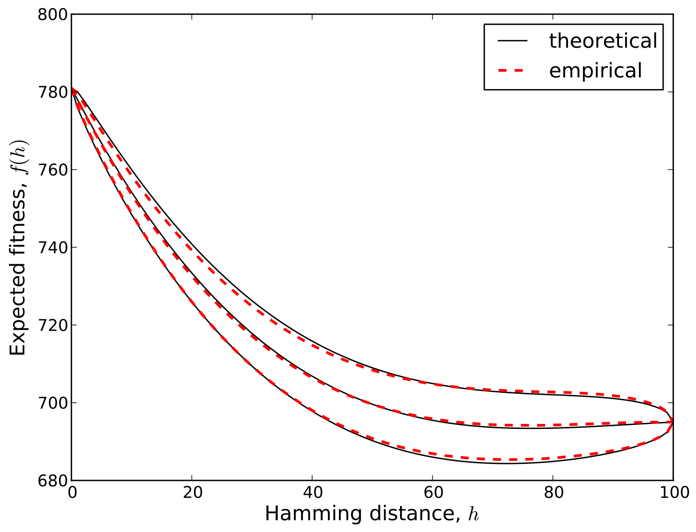


FIGURE C.2: Expected fitness in a Hamming-sphere of radius h. The mean fitness and the mean fitness plus and minus 1 standard deviation are shown. The empirical results are computed by sampling 10^6 configurations in each possible Hamming sphere.

As a final observation, we note that p_l is of order 1, while $s_l(\boldsymbol{x})$ is of order n, thus σ_2^2 is of order n and the fluctuations are of order $\sqrt{n}$.

Appendix D

Calculating the Cumulants for Graph-Colouring

The third cumulant for a graph G is given by,

$$K_3(G) = \mathbb{E}\left((c(G,\boldsymbol{x}) - \bar{c}(G))^3 \right) = \mathbb{E}\left(\left(\sum_{e \in \mathscr{E}} S_e \right)^3 \right).$$

We can represent the expansion of this term diagramatically. The variable S_e can be represented as a line between an edge of the graph. If we expand the cube of the sum we have terms consisting of three edge variables $S_e S_{e'} S_{e''}$ (note that the edges do not need to be different). These terms are represented by three edges in $\mathscr{G}$. If any edge connects a vertex not touched by the other edges then it will have zero expectation. Thus the only terms that contribute to $K_3(G)$ are those shown in Figure D.1.

Each triangle that occurs in the graph G occurs $3!$-times in the expansion of the cube of the sum. Thus the third cumulant is given by,

$$K_3(G) = |\mathscr{E}| \mathbb{E}\left(S_e^3 \right) + 3! |\mathscr{T}| \mathbb{E}\left(S_{(i,j)} S_{(j,k)} S_{(k,i)} \right),$$

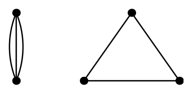


FIGURE D.1: This shows the two types of terms that contribute to the third cumulant.

where $\mathscr{T}$ is the set of three vertex cliques (i.e., triangles or loops of size 3) that exist in the graph G. It is straightforward, but tedious to show,

$$\begin{aligned}\mathbb{E}\left(S_e^3\right) &= \frac{1}{k}\left(1-\frac{1}{k}\right)\left(1-\frac{2}{k}\right)\\ \mathbb{E}\left(S_{(i,j)}\,S_{(j,k)}\,S_{(k,i)}\right) &= \frac{1}{k^2}\left(1-\frac{1}{k}\right).\end{aligned}$$

Thus,

$$K_3(G) = \frac{k^2-3k+2}{k^3}|\mathscr{E}| + \frac{3!(k-1)}{k^3}|\mathscr{T}|.$$

For graphs G drawn from $\mathscr{G}(n,p)$ the expected number of triangles in a graph is,

$$\mathbb{E}\left(|\mathscr{T}|\right) = \frac{p^3\,n\,(n-1)\,(n-2)}{3!}.$$

The diagrams that contribute to the fourth central moment are shown in Figure D.2. The terms that contribute to the fourth central moment are,

$$\begin{aligned}\mu_4(G) = \mathbb{E}\left((c(G,\boldsymbol{X})-\bar{c}(G))^4\right) &= \mathbb{E}\left(\left(\sum_{e\in\mathscr{E}} S_e\right)^4\right)\\ &= |\mathscr{E}|\left(\mathbb{E}\left(S_e^4\right)-3\mathbb{E}\left(S_e^2\right)^2\right)+\frac{3\times 4!}{2}|\mathscr{T}|\mathbb{E}\left(S_{(i,j)}^2\,S_{(j,k)}\,S_{(k,i)}\right)\\ &\quad +4!|\mathscr{S}|\mathbb{E}\left(S_{(i,j)}\,S_{(j,k)}\,S_{(k,l)}\,S_{l,i}\right)+3|\mathscr{E}|^2\mathbb{E}\left(S_e^2\right)^2,\end{aligned}$$

where $\mathscr{S}$ is the set of all loops of length four in the graph G. The combinatorial factors count the number of times each term occurs in the expansion. We note that in the last diagram we have a term $3\mathbb{E}\left(S_e^2\right)\mathbb{E}\left(S_{e'}^2\right)$ for all pairs of edges e and e' such that $e \neq e'$. By adding a term $3\,|\mathscr{E}|\mathbb{E}\left(S_e^2\right)^2$ to these diagrams we can write the total contribution as $3|\mathscr{E}|^2\mathbb{E}\left(S_e^2\right)^2$. In doing so we have over-counted the terms of order $|\mathscr{E}|$ by a factor $3\mathbb{E}\left(S_e^2\right)^2$ that we subtract from the first term. The term $3|\mathscr{E}|^2\mathbb{E}\left(S_e^2\right)^2$ is equal to $3K_2^2$ so that the fourth cumulant is given by,

$$\begin{aligned}K_4(G) &= \mu_4(G)-3K_2^2\\ &= |\mathscr{E}|\left(\mathbb{E}\left(S_e^4\right)-3\mathbb{E}\left(S_e^2\right)^2\right)+\frac{3\times 4!}{2}|\mathscr{T}|\mathbb{E}\left(S_{(i,j)}^2\,S_{(j,k)}\,S_{(k,i)}\right)\\ &\quad +4!|\mathscr{S}|\mathbb{E}\left(S_{(i,j)}\,S_{(j,k)}\,S_{(k,l)}\,S_{l,i}\right).\end{aligned}$$

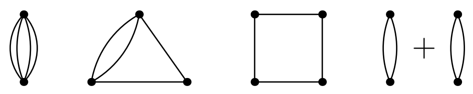


FIGURE D.2: This shows the four types of terms that contribute to the fourth central moment.

Evaluating the expectations we find,

$$\begin{aligned}
\mathbb{E}\left(S_e^4\right) &= \frac{1}{k}\left(1-\frac{1}{k}\right)\left(1-\frac{3}{k}+\frac{3}{k^2}\right)\\
\mathbb{E}\left(S_{(i,j)}^2\,S_{(j,k)}\,S_{(k,i)}\right) &= \frac{1}{k^2}\left(1-\frac{1}{k}\right)\left(1-\frac{2}{k}\right)\\
\mathbb{E}\left(S_{(i,j)}\,S_{(j,k)}\,S_{(k,l)}\,S_{l,i}\right) &= \frac{1}{k^3}\left(1-\frac{1}{k}\right).
\end{aligned}$$

Finally, the expected number of loops of 4 vertices for a graph drawn from $\mathscr{G}(n,p)$ is

$$\mathbb{E}\left(|\mathscr{S}|\right) = \frac{p^4\,n\,(n-1)\,(n-2)^2}{8}.$$

Bibliography

Jarmo T. Alander. *Practical Handbook of Genetic Algorithms: Complex Coding Systems, Chapter 13, Population size, building blocks, fitness landscape and genetic algorithm search efficiency in combinatorial optimization: An empirical study*, volume 3. CRC press, 1999.

J.T. Alander, L.A. Zinchenko, and S.N. Sorokin. Analysis of fitness landscape properties for evolutionary antenna design. In *Artificial Intelligence Systems, 2002. (ICAIS 2002). 2002 IEEE International Conference on*, pages 363 – 368, 2002.

AA Albrecht, P.C.R. Lane, and K. Steinh
"ofel. Analysis of Local Search Landscapes for k-SAT Instances. *Mathematics in Computer Science*, 3(4):465–488, 2010. ISSN 1661-8270.

L. Altenberg. Fitness distance correlation analysis: An instructive counterexample. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 57–64, San Mateo, CA, 1997. Morgan Kaufmann.

E Angel and V Zissimopoulos. Autocorrelation coefficient for the graph bipartitioning problem. *Theoretical Computer Science*, 191:229–243, 1998.

Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, September 1998. ISSN 0004-5411.

W. Benfold, J. Hallam, and A. Prügel-Bennett. Optimal parameters for search using a barrier tree Markov model. *Theoretical Computer Science*, 386, 2007.

K. D. Boese. Cost versus distance in the travelling salesman problem. Technical report, UCLA computer science department, Los Angeles, 1995.

K.D. Boese, A.B. Kahng, and S. Muddu. On the big valley and adaptive multi-start for discrete global optimizations. *Operation Research Letters*, 16(2), 1994a.

K.D. Boese, A.B. Kahng, and S. Muddu. On the big valley and adaptive multi-start for discrete global optimizations. *Operation Research Letters*, 16(2), 1994b.

H. Bouziri, K. Mellouli, and E. G. Talbi. Fitness Landscape Analysis for Optimum Multiuser Detection Problem. *Journal of Combinatorial Optimization*, 21(3):306–329, 2009.

Hend Bouziri, Khaled Mellouli, and El-Ghazali Talbi. The k-coloring fitness landscape. *Journal of Combinatorial Optimization*, 21:306–329, 2011. ISSN 1382-6905.

Jill Cirasella, David S. Johnson, Lyle A. McGeoch, and Weixiong Zhang. The asymmetric traveling salesman problem: Algorithms, instance generators, and tests. In *Revised Papers from the Third International Workshop on Algorithm Engineering and Experimentation*, ALENEX '01, pages 32–59. Springer-Verlag, 2001. ISBN 3-540-42560-8.

Philippe Collard, Sébastien Vérel, and Manuel Clergue. Local search heuristics: Fitness cloud versus fitness landscape. *CoRR*, abs/0709.4010, 2007.

S. A. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *Journal of the ACM*, 18(1):4–18, January 1971.

J. Culberson and I. Gent. Frozen development in graph coloring. *Theor. Comput. Sci.*, 265: 227–264, August 2001.

J. Czogalla. Fitness landscape analysis for the continuous flow-shop scheduling problem. In *Proceedings of 3rd European Workshop, Evo*, Naples, 2008.

J. Czogalla and A. Fink. *Fitness Landscape Analysis for the Resource Constrained Project Scheduling Problem*. Lecture Notes in Computer Science **5851**. Springer, Berlin, 2009.

Jens Czogalla and Andreas Fink. Fitness landscape analysis for the no-wait flow-shop scheduling problem. *Journal of Heuristics*, pages 1–27, 2011. ISSN 1381-1231.

C. Fonlupt, D. Robilliard, and P. Preux. Fitness landscape and the behavior of heuristics. In *in Evolution Artificielle 97 (EA'97*, 1997.

Stephanie Forrest and Melanie Mitchell. What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. Technical Report 3, 1993.

P. Galinier and J. K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.

Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

J. Garnier and L. Kallel. Efficiency of local search with multiple local optima. *SIAM J. Discr. Math., Vol*, 15:122–141, 2000.

C. A. Glass and A. Prügel-Bennett. A polynomially searchable exponential neighbourhood for graph colouring. *Journal of the Operational Research Society*, 56(3):324–330, 2005.

L. K. Grover. Local search and the local structure of NP-complete problems. *Operations Research Letters*, 12:235–243, 1992.

J. Hallam and A. Prügel-Bennett. Large barrier trees for studying search. *IEEE Transaction on Evolutionary Computation*, 9(4):385–397, 2005.

J. P. Hamiez and J. K. Hao. An analysis of solution properties of the graph coloring problem. In *4th metaheuristics international conference*, Porto, Portugal, 2001.

A. Hertz, B. Jaumard, and M. P. de Aragão. Local optima topology for the k-coloring problem. *Discrete Appl. Math.*, 49:257–280, 1994a. ISSN 0166-218X.

A. Hertz, B. Jaumard, and M Poggi de Aragao. Local optima topology for k-coloring problem. *Discrete Applied Mathematics*, 49:257–280, 1994b.

J Horn and D E Goldberg. Genetic algorithm difficulty and the modality of fitness landscapes. *Foundations of Genetic Algorithms*, 3, 1995.

Gregory Scott Hornby. The recombination operator, its correlation to the fitness landscape and search performance. In *Master of Science Thesis*. University of Alberta, 1996.

Tsutomu Hoshino, Daisuke Mitsumoto, and Tohru Nagano. Fractal fitness landscape and loss of robustness in evolutionary robot navigation. *Auton. Robots*, 5:199–213, 1998.

Dong Huang, Zhiqi Shen, Chunyan Miao, and Cyri Leung. Fitness landscape analysis for resource allocation in multiuser ofdm based cognitive radio systems. *SIGMOBILE Mob. Comput. Commun. Rev.*, 13:26–36, September 2009. ISSN 1559-1662.

D. Huanga, Z. Shenb, C. Miaoa, and C. Leungc. Fitness Landscape Analysis for Resource Allocation in Multiuser OFDM Based Cognitive Radio Systems. *Mobile Computing and Communications Review*, 13(2):26–36, 2009.

T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, 1995a.

T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, 1995b.

R. Lefticaru and F. Ipate. A comparative landscape analysis of fitness functions for search-based testing. In *IEEE 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, USA, 2008.

Remi Lehn and Pascale Kuntz. A contribution to the study of the fitness landscape for a graph drawing problem. In Egbert Boers, editor, *Applications of Evolutionary Computing*, volume 2037 of *Lecture Notes in Computer Science*, pages 172–181. Springer Berlin / Heidelberg, 2001. ISBN 978-3-540-41920-4.

Guanzhou Lu, Jinlong Li, and Xin Yao. Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms. In *Proceedings of the 11th European conference on Evolutionary computation in combinatorial optimization*, EvoCOP'11, pages 108–117, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-20363-3.

B. Manderick, M. de Weger, and P. Spiessens. The genetic algorithm and the structure of the fitness landscape. In *Proceedings of 4th International Conference on Genetic Algorithms*, pages 143–150, 1991.

W.N. Martin, A.L. Barker, and J.P. Cohoon. Problem perturbation: implications on the fitness landscape. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, 1999.

Keith Mathias and Darrell Whitley. Genetic operators, the fitness landscape and the traveling salesman problem. In *Parallel Problem Solving from Nature*, pages 219–228. Elsevier Science Publishers, 1992a.

Keith Mathias and Darrell Whitley. Genetic operators, the fitness landscape and the traveling salesman problem. In *Parallel Problem Solving from Nature*, pages 219–228. Elsevier Science Publishers, 1992b.

I. P. McCarthy. Manufacturing strategy: understanding the fitness landscape. *International Journal of Operations and Production Management*, 24(2):124–150, 2008.

P. Merz and B. Freisleben. *Memetic Algorithms and the Fitness Landscape of the Graph Bi-Partitioning Problem*, volume 1498 of *Lecture Notes in Computer Science*. Springer, Berlin, 1998a.

P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *Evolutionary Computation, IEEE Transactions on*, 4(4):337 – 352, November 2000. ISSN 1089-778X.

Peter Merz. Advanced fitness landscape analysis and the performance of memetic algorithms. *Evol. Comput.*, 12:303–325, 2004.

Peter Merz and Bernd Freisleben. Memetic algorithms and the fitness landscape of the graph bi-partitioning problem. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, PPSN V, pages 765–774. Springer-Verlag, 1998b. ISBN 3-540-65078-4.

D Mitchell, B Selman, and H Levesque. Hard and easy distributions of {SAT} problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 459–465, San Jose, CA, USA, 1992. Publ by AAAI, Menlo Park, CA, USA.

P. Moscato. On evolution, search, optimisation, genetic algorithms and martial arts: Toward memetic algorithms. Technical report, California Institute of Technology, Pasadena, 1989.

P. Moscato and M. G. Norman. A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimisation on message-passing systems. In *Proceedings of the International Conference on Parallel Computing and Transputer Applications*, pages 177–186, 1992.

D. Newth and M. Brede. Fitness Landscape Analysis and optimisation of Coupled Oscillators. *Journal of Complex Systems*, 16:317–331, 2006.

Petr Pošík and Vojtěch Franc. Estimation of fitness landscape contours in eas. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 562–569, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computation*. Cambridge University Press, Cambridge, UK., 3rd edition, 2007.

A. Prügel-Bennett. On the chromatic number of finite random graphs, journal =.

Adam Prügel-Bennett and M.-H. Tayarani-N. Maximum satisfiability: Anatomy of the fitness landscape for a hard combinatorial optimisation problem. *IEEE Transactions on Evolutionary Computation*, 16(3):319–338, 2011.

M. Qasem and A. Prügel-Bennett. Complexity of max-sat using stochastic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008*, pages 615–616. ACM, 2008.

M. Qasem and A. Prügel-Bennett. Learning the large-scale structure of the max-sat landscape using populations. *IEEE Transactions on Evolutionary Computation*, 14(4):518–529, 2010.

R. L. Rardin, C. Tovey, M. Pilcher, and P. Pardalos. Analysis of a random cut test instance generator for the tsp. In *World Scientific*, 1993.

Alain Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, PPSN V, pages 87–96, London, UK, 1998. Springer-Verlag. ISBN 3-540-65078-4.

C. R. Reeves and A. V. Eremeev. Statistical analysis of local search landscapes. *The Journal of the Operational Research Society*, 55(7):pp. 687–693, 2004. ISSN 01605682.

Colin R. Reeves and Mériéma Aupetit-Bélaidouni. Estimating the number of solutions for sat problems. In *PPSN*, pages 101–110, 2004.

J. Riley and V. Ciesielski. Fitness landscape analysis for evolutionary non-photorealistic rendering. In *Proceedings of IEEE World Congress on Computational Intelligence*, Barcelona, 2010.

Bart Selman, Hector J. Levesque, and David G. Mitchell. A new method for solving hard satisfiability problems. In *AAAI*, pages 440–446, 1992.

W. Shaowei and Z. Qiuping. Fitness Landscape Analysis for Optimum Multiuser Detection Problem. *Journal of Natural Sciences*, 12(6):1073–1076, 2007.

Liang Shen and Jun He. A mixed strategy for evolutionary programming based on local fitness landscape. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –8, july 2010.

K. Slany and L. Sekanina. Fitness landscape analysis and image filter evolution using functional-level cgp. In *Proceedings of the 10th European conference on Genetic programming*, pages 311–320, 2007.

P. Stadler. Towards a theory of landscapes. *Complex Systems and Binary Networks*, pages 78–163, 1995.

P. Stadler. Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, 20 (1):1–45, 1996.

Peter F. Stadler and Wolfgang Schnabl. The landscape of the traveling salesman problem. *Physics Letters A*, 161(4):337 – 344, 1992. ISSN 0375-9601.

A. M. Sutton, A. E. Howe, and L. D. Whitley. A theoretical analysis of the k-satisfiability search space. In *Proceedings of the Second International Workshop on Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, SLS '09, pages 46–60, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-03750-4.

H. Suzuki and Y. Iwasa. Ga performance in a babel-like fitness landscape. In *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*, pages 357–366, 1997.

J. Tavares, F. B. Pereira, , and E. Costa. Multidimensional Knapsack Problem: A Fitness Landscape Analysis. *IEEE Transactions on Systems, Man, and Cybernetics -Part B*, 38(3): 604–616, 2008.

J. Tavares, F.B. Pereira, and E. Costa. The role of representation on the multidimensional knapsack problem by means of fitness landscape analysis. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2307 –2314, 0-0 2006.

M. H. Tayarani-N and Adam Prügel Bennett. Anatomy of the fitness landscape for graph-colouring problem. *accepted with revision in Journal of Evolutionary Computation, MIT Press*, 10:1, 2011.

M. H. Tayarani-N and Adam Prügel Bennett. On the landscape of combinatorial optimisation problems. *accepted with revision in IEEE Transactions on Evolutionary Computation*, 10:1, 2012a.

M. H. Tayarani-N and Adam Prügel Bennett. Quadratic assignment problem: A landscape analysis. *Submitted to IEEE Transactions on System, Man and Cybernetics*, 10:1, 2012b.

M. H. Tayarani-N and Adam Prügel Bennett. Travelling salesman problem: A landscape analysis. *accepted with revision in Journal of Evolutionary Computation, MIT Press*, 10: 1, 2012c.

Leonardo Vanneschi, Marco Tomassini, Philippe Collard, Sbastien Vrel, Yuri Pirola, and Giancarlo Mauri. A comprehensive view of fitness landscapes with neutrality and fitness clouds. In *Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, pages 241–250. Springer Berlin - Heidelberg, 2007.

S. Vérel, P. Collard, M. Tomassini, and L. Vanneschi. Fitness landscape of the cellular automata majority problem: View from the "olympus". *Theor. Comput. Sci.*, 378:54–77, 2007.

Sébastien Vérel, Philippe Collard, Marco Tomassini, and Leonardo Vanneschi. Neutral fitness landscape in the cellular automata majority problem. *CoRR*, abs/0803.4240, 2008.

Jean-Paul Watson. An introduction to fitness landscape analysis and cost models for local search. *HANDBOOK OF METAHEURISTICS*, 146:599–623, 2010.

E. D. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biol. Cyber.*, 63:325–336, 1990.

Weixiong and Zhang. Configuration landscape analysis and backbone guided local search.: Part i: Satisfiability and maximum satisfiability. *Artificial Intelligence*, 158(1):1 – 26, 2004. ISSN 0004-3702.

S. Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proceedings of 6th Congress of Genetics*, volume 1, page 365. ACM Press, 1932.

Yanghui Wu, J. McCall, and D. Corne. Fitness landscape analysis of bayesian network structure learning. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 981 –988, june 2011.

H. Yoshizawa and S. Hashimoto. Landscape analyses and global search of knapsack problems. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 3, pages 2311 –2315 vol.3, 2000.

W. Zhang, A. Rangan, and M. Looks. Backbone guided local search for maximum satisfiability. In *Proc. of the 18th Intern. Joint Conference on Artifical Intelligence*, pages 1179–84, 2003.

Weixiong Zhang. Configuration landscape analysis and backbone guided local search: part i: Satisfiability and maximum satisfiability. *Artif. Intell.*, 158:1–26, September 2004. ISSN 0004-3702.