OPTIMIZATION BY QUANTUM ANNEALING FOR THE GRAPH COLOURING PROBLEM

OLAWALE TITILOYE

A thesis submitted in partial fulfilment of the requirements of the Manchester Metropolitan University for the degree of Doctor of Philosophy

School of Computing, Mathematics and Digital Technology

March 2013

Abstract

Quantum annealing is the quantum equivalent of the well known classical simulated annealing algorithm for combinatorial optimization problems. Despite the appeal of the approach, quantum annealing algorithms competitive with the state of the art for specific problems hardly exist in the literature. Graph colouring is a difficult problem of practical significance that can be formulated as combinatorial optimization. By introducing a symmetry-breaking problem representation, and finding fast incremental techniques to calculate energy changes, a competitive graph colouring algorithm based on quantum annealing is derived. This algorithm is further enhanced by tuning simplification techniques; replica spacing techniques to increase robustness; and a messaging protocol, which enables quantum annealing to efficiently take advantage of multiprocessor environments. Additionally, observations of some patterns in the tuning for random graphs led to a more effective algorithm able to find new upper bounds for several widely-used benchmark graphs, some of which had resisted improvement in the last two decades.

Acknowledgements

I am immensely grateful to my supervisors Dr Alan Crispin and Dr Zuhair Bandar for their help during the course of my research. I thank the Dalton Research Institute of the Manchester Metropolitan University for funding the research presented in this thesis. I also thank my parents, the rest of my family and my friends for their support.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background on Quantum Annealing

A combinatorial optimization problem can be considered as the search for the global minimum of a multi-variable cost function. Combinatorial optimization problems of large sizes often require algorithms based on stochastic local search [Hoos & Stiitzle, 2004]. Classical (simulated) Annealing (CA) [Kirkpatrick et al., 1983; Černý, 1985] is a well known stochastic local search algorithm that employs a temperature parameter to overcome barriers in the potential energy landscape defined by the cost function being addressed. Quantum Annealing (QA) [Finnila et al., 1994; Kadowaki & Nishimori, 1998] is an analogous algorithm based on quantum mechanics. In QA, the main control parameter is a magnetic field strength, which introduces artificial quantum fluctuations for the purpose of tunnelling through barriers, instead of having to thermally overcome them.

Problem instances of very small sizes can be approached by a variant of QA that follows the evolution of the Schrödinger equation, (the characteristic equation of quantum mechanics) by numerical integration [Stella et al., 2005]. This variant is very similar to another approach termed quantum adiabatic evolution presented in [Farhi et al., 2001], where only problems of very small sizes were also investigated. Unless an actual scalable quantum computer is available, exact quantum adiabatic evolution is known to be intractable for problem sizes with practical significance [Farhi et al., 2001]. Although the components of a scalable adiabatic quantum computer are being investigated [Johnson et al., 2011], the research is still in its early stages. Therefore, approximate stochastic formulations with quantum Monte Carlo are important for large problem sizes. Path Integral Monte Carlo (PIMC) is one of the most promising quantum Monte Carlo approaches, with studies already carried out for many problems including protein modelling and Lennard-Jones clusters [Lee & Berne, 2000], the two-dimensional random Ising Model [Santoro et al., 2002; Martoňák et al., 2002], the travelling salesman problem [Martoňák et al., 2004], Boolean satisfiability [Battaglia et al., 2005], clustering [Kurihara et al., 2009] and Variational Bayesian inference [Sato et al., 2009].

Most of the existing results have concentrated on investigating whether quantum annealing can outperform classical annealing, even though in most cases classical annealing was not the best performing algorithm for the problem under consideration [Martoňák et al., 2004; Battaglia et al., 2005]. While this approach has provided useful insights, it is desirable to move forward and design quantum annealing algorithms that can match and even outperform the best alternatives for a given problem. The graph colouring problem is a fundamental hard optimization problem having practical applications such as scheduling, and had not been addressed in the context of quantum annealing before the commencement of this thesis, but has recently been attempted with negative conclusions in [Lecina, 2011]. The main aim is to present a competitive graph colouring algorithm based on quantum annealing.

1.2 Background on Graph Colouring

The graph colouring problem requires that we find the minimum number of colours with which the vertices of a given graph can be labelled so that the graph is properly coloured. This minimum number is known as the chromatic number of the graph. A graph is properly coloured if no vertices connected by an edge receive the same colour. The decision version of the graph colouring problem is known as the *k*-colouring problem, which involves deciding whether a given graph can be properly coloured when only *k* colours are available. The *k*-colouring problem was one of the first to be classified as NP-complete [Karp, 1972; Garey & Johnson, 1979]. An NP-complete problem is a decision problem for which a purported solution can be verified efficiently, but no efficient way is known in which a solution could be obtained in the first place for all instances of the problem. An algorithm is considered to be efficient if it completes in a running time bounded by a polynomial in the input size of problem [Cobham, 1965]. Moreover, if any efficient algorithm is found for one NP-complete problem, then every NP-complete problem can be efficiently solved [Cook, 1971].

The best exact algorithms for the graph colouring problem only terminate in reasonable time for small input sizes of about a hundred vertices [Malaguti et al., 2011; San Segundo, 2012; Gualandi et al., 2012]. Over the years, several heuristic approaches that relax the problem by approximations, or by forgoing guarantees that the exact chromatic number is found have been investigated. These include greedy construction [Brélaz, 1979; Leighton, 1979], classical simulated annealing [Chams et al., 1987; Johnson et al., 1991; Morgenstern, 1996], Tabu search [Hertz & Werra, 1987] and Evolutionary-Tabu hybrids [Fleurent & Ferland, 1996; Galinier & Hao, 1999].

As well as being a widely used problem for testing new concepts in the development of metaheuristics, graph colouring has many important practical applications including scheduling and timetabling [Leighton, 1979; Mehta, 1981; Sabar et al., 2012], register allocation [Chaitin, 1982; Chow & Hennessy, 1984; Briggs et al., 1989], frequency assignment [Hale, 1980; Dorne & Hao, 1995], printed circuit testing [Garey et al., 1976], and computing derivatives [Gebremedhin et al., 2005; Hossain & Steihaug, 2008].

1.3 Thesis Outline

This thesis consists of seven chapters, including this introductory chapter. Chapter 2 contains a review of quantum annealing as already applied to a number of combinatorial optimization problems. The main emphasis is on quantum Monte Carlo, as this is the only feasible way to implement quantum annealing on a classical computer for problems of large sizes. We specifically focus on Path Integral Monte Carlo (PIMC), which has been demonstrated to be a promising implementation of quantum Monte Carlo [Das & Chakrabarti, 2008].

Chapter 3 consists of a review of approaches to the graph colouring problem, with an emphasis on heuristics rather than exact algorithms, which can only solve small problem instances. Greedy construction algorithms, as well as local search and population based approaches are reviewed.

In chapter 4, we introduce a quantum annealing algorithm for the graph colouring problem. An intricate problem representation that avoids graph independent symmetry problems arising from the redundancy of colour naming is presented. Alongside this, fast incremental techniques are developed in order to calculate cost function changes that arise from using the newly introduced graph colouring representation. These lead to a competitive graph colouring algorithm which is able to find a chromatic number upper bound previously unknown in the literature.

Chapter 5 is concerned with enhancements and variations to the quantum annealing algorithm presented in chapter 4. Parameter tuning is simplified after the observation that it is not necessary to continually decrease the value of the field strength like in the standard version. A replica spacing technique inspired by evolutionary algorithms is also introduced to combat premature convergence and improve the effectiveness of quantum annealing.

Additionally a messaging protocol is presented to enable quantum annealing to take advantage of multi-processor environments efficiently. These changes lead to a more powerful graph colouring algorithm, which found additional improvements for benchmark graphs used in the literature.

In chapter 6, tuning patterns for random graphs are investigated. Useful observations are made in relation to how the evolution of the move acceptance ratio relates to the success of the algorithm. These lead to a tuning mechanism that helps significantly in the solving of some difficult k-colouring problem instances that had been open for about two decades. Chapter 7 contains the conclusions.

1.4 Contributions

The main contributions can be summarized as follows:

- The first successful quantum annealing algorithm for graph *k*-colouring, featuring an effective Boolean representation, and fast incremental calculations for the cost function
- Enhancement of the quantum annealing *k*-colouring algorithm by incorporating tuning simplification, replica spacing techniques, and parallelization
- Insights into the optimal tuning of quantum annealing for the *k*-colouring of random graphs by exploiting an observation in which certain parameter values lead to a continuously increasing acceptance ratio

In more detail, the contributions are the following:

An initial quantum annealing algorithm for graph *k*-colouring [chapter 4]: In addition to the cost function or potential energy used in a classical annealing algorithm, quantum annealing requires a kinetic energy defined in terms of a

Boolean variable representation of the problem. In the case of the graph k-colouring problem, the usual potential energy to be minimized is the number of conflicting edges in the graph. We identify a Boolean representation which exploits some problem specific characteristics of graph colouring. As the overall cost function used in quantum annealing is more complex due to the inclusion of a kinetic energy, fast incremental techniques to calculate changes to the total cost function are imperative. These are found. The resultant graph colouring algorithm is competitive with the best ones in the literature on a set of well known benchmark graphs. In particular, the presented quantum annealing algorithm became the first to successfully colour the graph DSJC1000.9 with 222 colours.

Enhancing quantum annealing by fixing Gamma, spacing the replicas, and specifying a parallelization approach [chapter 5]: Gamma, the quantum fluctuation parameter, is traditionally lowered gradually from a high value throughout the course of the Monte Carlo simulation. It is experimentally demonstrated that for graph colouring, heuristically setting Gamma to a fixed value yields an improved and more easily managed algorithm. Quantum annealing maintains a population of separate configurations of the problem instance called replicas. Due to large computational requirements, only a small number of replicas can be used in practice in a Monte Carlo algorithm. This can lead to replicas prematurely converging and getting trapped by poor local optima. Because evolutionary algorithms have to deal with similar issues, some of their techniques such as population spacing and mutations can be applicable. This thesis incorporates them into quantum annealing for the first time, thereby resulting in a more robust graph colouring algorithm. We also present a scheme whereby quantum annealing can take advantage of multiple processors, with replicas acting as agents that occasionally send messages to each other. For the first time in the literature, 97-colourings were discovered for the widely studied benchmark Latin square graph, Latin_square_10.

Tuning quantum annealing to produce a continuously increasing acceptance ratio for random graphs [chapter 6]: It is experimentally demonstrated for dense random graphs that the best tuning parameter values for the most difficult instances are those that cause the acceptance ratio to diverge over time. This result led to the solution of several well known *k*-colouring benchmark problems, some of which had been open for almost two decades. These include 47-colourings and 82-colourings, obtained for the first time in the literature, for the well studied DSJC500.5 and DSJC1000.5 random graphs respectively.

1.5 Publications

1. Titiloye, O., & Crispin, A. (2011). Quantum annealing of the graph coloring problem. *Discrete Optimization*, *8*(2), 376-384.
2. Titiloye, O., & Crispin, A. (2011). Graph coloring with a distributed hybrid quantum annealing algorithm. *Agent and Multi-Agent Systems: Technologies and Applications*, 553-562.
3. Titiloye, O., & Crispin, A. (2012). Parameter Tuning Patterns for Random Graph Coloring with Quantum Annealing. *PloS one*, *7*(11), e50060.

The first paper is the journal publication listed in the references section as [Titiloye & Crispin, 2011a] on which chapter 4 is based; the second paper is the conference paper listed as [Titiloye & Crispin, 2011b] on which chapter 5 is based; and the third paper is the journal paper listed as [Titiloye & Crispin, 2012] on which chapter 6 is based.

Different stages of this research produced several graph colouring upper bounds new to the literature for widely-used benchmark graphs. Some of these were also found contemporaneously and independently by [Wu & Hao, 2012] and [Hao & Wu, 2012] using

Evolutionary-Tabu and set extraction algorithms. The chronology of the results is preserved by presenting them in the chapters that correspond to our publications.

Chapter 2

Combinatorial Optimization by Quantum Annealing

Many real world problems can be formulated in terms of combinatorial optimization. In the difficult cases where efficient exact algorithms for the problem at hand are unknown, a useful compromise is to seek for solutions or approximate solutions using stochastic local search. The stochastic local search algorithm under consideration is the Path Integral Monte Carlo (PIMC) implementation of Quantum Annealing (QA) [Santoro et al., 2002; Martoňák et al., 2002; Das & Chakrabarti, 2008], which we refer to as PIMC-QA or simply QA or quantum annealing, when it is clear from the context. In what follows, the standard formulation of PIMC-QA is reviewed in the context of the Ising model in Section 2.1, Boolean satisfiability in Section 2.2 and the travelling salesman problem in Section 2.3.

2.1 An Ising Optimization Formulation

Quantum annealing seeks to exploit an analogy between a combinatorial optimization problem and the physics of a many-body problem to which quantum mechanics is applicable. Therefore the desired formulation of combinatorial optimization in this context is that of an Ising Model [Cipra, 1987]. The Ising Model was introduced by [Ising, 1925] in order to study the properties of collections of small interacting particles. Many combinatorial optimization problems can be cast into a form similar to the Ising Model or a related multi-state Potts spin model [Das & Chakrabarti, 2008]. Finding the ground state (or global minimum) of an Ising model is NP-hard in the general case [Barahona, 1982]. In the Ising spin glass problem, we are presented with a system having a set of N spins $S = \{S_1, S_2, \dots, S_N\}$, each of which can take one of two possible values ± 1. The spins are considered to have interactions with their nearest neighbours, and the problem is to minimize the total energy or Hamiltonian given as

$$\mathcal{H}_p = -\sum_{\langle ij \rangle} J_{ij} S_i S_j \tag{2.1}$$

The numbers J_{ij} represent the strength of the interactions. A positive value for J_{ij} is considered ferromagnetic (or attracting), while a negative value is anti-ferromagnetic (or repelling). The expression $\langle ij \rangle$ signifies that the pairs (ij) and (ji) are counted only once. The system in (2.1) is known as the Edwards-Anderson model of the Ising spin glass [Edwards & Anderson, 1975, Pál, 1996, Marinari et al., 1998].

When studying the dynamics of an Ising model, a very important statistical mechanics quantity is the partition function Z, which is useful for several things including calculating the probability of being in a given configuration [Thompson, 1979]. The partition function is defined as

$$Z = \sum_{\pm 1} \exp(-\beta \mathcal{H}_{\mathrm{p}}) \tag{2.2}$$

The expression for Z in (2.2) is the sum of the exponentiation of the Hamiltonian over all of the 2^N configurations in which each spin is assigned either $+1$ or -1. The quantity β is given by $1/k_B T$, where T is the temperature and k_B is the Boltzmann constant, which is usually set to 1. Instead of using the summation symbol Σ, Z is often written as $\mathrm{Tr} \exp(-\beta \mathcal{H}_p)$, where

Tr is the word trace symbol. The probability that the system is in a particular configuration S is then given by

$$Prob(S) = \frac{\exp(-\beta\mathcal{H}_p(S))}{Z} \tag{2.3}$$

Computations for deriving properties of physical systems were first performed by [Metropolis et al., 1953] with what is now known as the Metropolis algorithm. This was later extended to general combinatorial optimization problems [Kirkpatrick et al., 1983; Černý, 1985] in a process now widely known as classical simulated annealing (CA). The quantities in (2.2) and (2.3) play a part in the derivation of CA.

Essentially, a random change is made to one of the spins during each move. If the change in the Hamiltonian $\Delta\mathcal{H}_p$ is negative, the new configuration is accepted. Otherwise, the new configuration is only accepted with a probability of $\exp(-\Delta\mathcal{H}_p/T_t)$, known as the Metropolis criterion, where T_t is the temperature at time t.

Quantum annealing (QA) attempts to improve on CA by using quantum fluctuations to tunnel through barriers to the ground state [Das & Chakrabarti, 2008], rather than, or in addition to overcoming the barriers thermally. When a search process encounters a local minimum, then without the availability of tunnelling, the process would have to visit configurations with costs higher than the barrier in order to escape. Any mechanism that facilitates an escape from local minimum without having to scale this height can be described as tunnelling. This is what QA attempts to achieve as illustrated in [Das & Chakrabarti, 2008]. For computational feasibility reasons, we are interested in Monte Carlo implementations of QA, which can be approximately simulated on a classical computer. One of the most promising versions of Monte Carlo QA for large problem instances is the Path Integral Monte Carlo Quantum annealing (PIMC-QA) [Santoro et al., 2002; Martoňák et al., 2002; Martoňák et al., 2004]. In the main version of PIMC-QA, a kinetic energy term is defined as some property of the system that does not commute with the potential energy. A magnetic field strength term Γ is also introduced to control the amount of influence that the kinetic energy has on the system in the presence of a small temperature T. The value of the field strength Γ is then lowered from a suitably high value towards zero, in an attempt to anneal the system towards its classical ground state.

In order to formulate PIMC-QA, the classical Hamiltonian is expressed as a quantum Hamiltonian in a transverse magnetic field providing quantum fluctuations [Santoro et al., 2002; Martoňák et al., 2002]. In the case of the Ising spin glass problem, the Hamiltonian $\mathcal{H}_p$ in (2.1) is re-written as

$$\mathcal{H}_q = -\sum_{\langle ij \rangle} J_{ij} \sigma_i^z \sigma_j^z - \Gamma \sum_i \sigma_i^x \tag{2.4}$$

The symbols σ_i^x and σ_i^z are Pauli matrices for the spin i in the x and z directions respectively. Pauli matrices are a set of two by two matrices known to be important in quantum mechanics. Specifically, $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. The derivation of a path integral formulation for the quantum Ising model in (2.4) is known as a Suzuki-Trotter transformation of a d-dimensional quantum Hamiltonian into a $d + 1$ dimensional classical Hamiltonian [Suzuki, 1976; Das & Chakrabarti, 2008]. The Suzuki-Trotter transformation is a way of expressing a quantum Hamiltonian as a new classical Hamiltonian, thereby making it easier to perform Monte Carlo simulations on a classical computer. In order to apply a Suzuki-Trotter transformation, the quantum Hamiltonian $\mathcal{H}_q$ in (2.4) is re-written as the sum of two parts $\mathcal{H}_q = \mathcal{H}_{pot} + \mathcal{H}_{kin}$, where $\mathcal{H}_{pot} = -\sum_{\langle ij \rangle} J_{ij} \sigma_i^z \sigma_j^z$, and $\mathcal{H}_{kin} = -\Gamma \sum_i \sigma_i^x$. The partition function is then expressed as $Z = \mathrm{Tr}\exp(-(\mathcal{H}_{kin} + \mathcal{H}_{pot})/T)$. The Trotter formula [Trotter, 1959], is given by $\exp(A_1 + A_2) = \lim_{P\to\infty}[\exp A_1/P \exp A_2/P]^M$. This gives $Z = \lim_{P\to\infty} \sum_i \langle S_i| \left[\exp(-\mathcal{H}_{kin}/PT) \times \exp(-\mathcal{H}_{pot}/PT)\right]^P |S_i\rangle$. After further algebraic manipulation, which can be found in [Martoňák et al., 2002, Das & Chakrabarti, 2008], the quantum partition function is decomposed into a classical partition function consisting of P interacting replicas, thereby deriving a new classical Hamiltonian, which can be written with notation from [Battaglia et al., 2005] as

$$\mathcal{H} = -\frac{1}{P}\sum_{\rho=1}^{P}\sum_{\langle ij\rangle} J_{ij} S_{i,\rho} S_{j,\rho} - J_\Gamma \sum_{\rho=1}^{P}\sum_{i} S_{i,\rho} S_{i,\rho+1} \tag{2.5}$$

where $S_{i,\rho}$ is the ith spin of the ρth replica, and J_Γ is given by

$$J_\Gamma = -\frac{T}{2}\ln\tanh\left(\frac{\Gamma}{PT}\right) > 0 \tag{2.6}$$

The result of the Suzuki-Trotter transformation in (2.5) is based on the assumption that a single spin flip is sufficient to take a valid configuration of the problem to another valid configuration [Das & Chakrabarti, 2008]. This is true for the Ising spin glass problem [Martoňák et al., 2002]. Quantum annealing can now proceed as a simulation of $\mathcal{H}$ in a Metropolis algorithm as performed in [Santoro et al., 2002; Martoňák et al., 2002].

2.2 The Application of QA to Boolean Satisfiability

In this section, we review a study in which a Boolean satisfiability problem was approached with QA [Battaglia et al., 2005]. Boolean satisfiability is concerned with whether a Boolean formula can be satisfied, and is applicable to several real-world problems including circuit design and automated planning [Marques-Silva, 2008]. The particular type of Boolean satisfiability problem under consideration is 3-SAT, which is one of the most fundamental NP-complete problems [Cook, 1971].

In the 3-SAT problem, we have a set of N Boolean variables $\{x_1, x_2, \dots, x_n\}$. Each variable x_i can only take a value of either 1(TRUE) or 0(FALSE). A 3-SAT formula is a conjunction of 3-clauses, which are themselves disjunctions of three selections from the Boolean variables and their negations. A conjunction refers to the use of the logical AND ($\wedge$), while a disjunction is the use of the logical OR ($\vee$). An example of a clause is $C = (x_5 \vee \neg x_7 \vee x_8)$, where $\neg x_i$ means that the value of x_i is negated in the clause. A 3-SAT problem consists of determining whether there exists an assignment of $\{x_i\}$ to values such that a given formula with M clauses $C_1 \wedge C_2 \dots \wedge C_M$ evaluates to TRUE.

The 3-SAT problem can be mapped to an Ising model by setting $S_i = 2x_i - 1$, and defining a classical Hamiltonian

$$\mathcal{H}_p(\{S_i\}) = \sum_{a=1}^{M} \frac{(1 + J_{a,i}S_i)(1 + J_{a,j}S_j)(1 + J_{a,k}S_k)}{8} \qquad (2.7)$$

which is the sum over the evaluation of each of the M clauses. The value $J_{a,i}$ is defined to be -1 if the Boolean variable x_i associated with the spin variable S_i appears negated in the clause, and $+1$ otherwise. The Hamiltonian $\mathcal{H}_p$ in (2.7) counts the number of violated clauses, and is therefore equal to zero if and only if a satisfying assignment has been found. A 3-SAT formula can originate from a practical application, or it can be generated from a probability distribution for the purpose of experiments. The instance of 3-SAT considered in [Battaglia et al., 2005] belongs to the distribution of the uniform random 3-SAT [Hoos &

Stiitzle, 2000]. In the uniform random 3-SAT, each clause is generated by selecting three variables out of the available N at random. Each variable either appears as is, or negated with uniform probability. In order to apply QA to the 3-SAT problem, the classical Hamiltonian in (2.7) is first expressed as a quantum Hamiltonian in a transverse field, just as in the case of the Ising spin glass problem in Section 2.1. Such a quantum Hamiltonian can be written as

$$\mathcal{H}_q = \mathcal{H}_p(\{\sigma_i^z\}) - \Gamma \sum_i \sigma_i^x \tag{2.8}$$

The abbreviated notation $\mathcal{H}_p(\{\sigma_i^z\})$ from [Battaglia et al., 2005] helps avoid having to write out the whole of the expression for $\mathcal{H}_p$ given in (2.7). As was the case for the Ising spin glass in Section 2.1, after a Suzuki-Trotter transformation of the Hamiltonian in (2.8), we arrive at a new classical Hamiltonian

$$\mathcal{H} = \frac{1}{P}\sum_{\rho=1}^{P} \mathcal{H}_p(\{S_{i,\rho}\}) - J_\Gamma \sum_{\rho=1}^{P}\sum_i S_{i,\rho}S_{i,\rho+1} \tag{2.9}$$

The Hamiltonian in (2.9) is analogous to the one in (2.5) and J_Γ is defined as in (2.6).

An application of CA to the 3-SAT problem consists of using the original Hamiltonian $\mathcal{H}_p$ in (2.7). Starting from a randomly initialized state, the CA process repeatedly selects a spin at random to be flipped, and the change $\Delta\mathcal{H}_p$ calculated. The new configuration is accepted if either $\Delta\mathcal{H}_p < 0$ or $\exp(-\Delta\mathcal{H}_\mathrm{p}/\mathrm{T}_\mathrm{t}) > r$, where r is a newly generated random number between 0 and 1, and T_t is the temperature at time t. The temperature is set to a sufficiently high initial value, and then gradually lowered over time according to an annealing schedule such as a linear or a geometric one [Strenski &

Kirkpatrick, 1991]. In the case of the application of QA to the 3-SAT problem, the Suzuki-Trotter transformed classical Hamiltonian $\mathcal{H}$ in (2.9) can be used in a Metropolis algorithm, which is like a simulated annealing process with a small fixed temperature. In contrast to the CA process which has N spins, the QA process has PN spins each of which is flipped uniformly at random. After a spin flip, the new configuration is accepted if $\exp(-\Delta\mathcal{H}/\mathrm{T}) > r$. Over time, the field strength Γ (on which $\mathcal{H}$ is dependent), is decreased linearly from a sufficiently high value towards zero as in [Santoro et al., 2002].

Experiments were carried in [Battaglia et al., 2005] with a uniform random 3-SAT instance having $N = 10^4$ variables. It was generated such that the value $\alpha = M/N$ was 4.2, and therefore very close to the unsatisfiability phase transition [Monasson et al., 1999; Mézard et al., 2002]. The formula used was first verified to be satisfiable using WALKSAT [Selman et al., 1993], a well known SAT solver. Even though QA did not compete very well with existing approaches to the 3-SAT, the study is very instructive on the stages to follow when applying QA to a combinatorial optimization problem.

2.3 The Application of QA to the Travelling Salesman Problem

The travelling salesman problem (TSP) is an NP-hard problem [Garey & Johnson, 1979] having practical applications, and widely used as a test bed for newly introduced search procedures. For example, the TSP was one of the first problems to be addressed when CA was introduced [Kirkpatrick et al., 1983; Černý, 1985]. It was therefore interesting to see how a PIMC-QA algorithm could be implemented for the problem, and what the performance would be. This study has been carried out [Martoňák et al., 2004], and will be reviewed in this section.

In the TSP, we are given a set of cities $\{c_1, c_2, \dots, c_n\}$ and an $n \times n$ matrix d, where d_{ij} contains the distance between city i and city j. We are required to find the shortest tour length that allows us to visit every city once and return to the starting point. Like many studies, [Martoňák et al., 2004] concentrates on the symmetric case where $d_{ij} = d_{ji}$. An arbitrary configuration can be represented by a permutation of cities. Therefore, there are a total of $(n-1)!$ possibilities. In order to represent a tour in terms of Boolean variables and then Ising spin variables, an $n \times n$ matrix T is defined, such that $T_{ij} = 1$ if city i directly follows city j in the tour, and $T_{ij} = 0$ otherwise. In this system, only n entries in T are non-zero. Another matrix U is also defined such that $U = T + T^t$, where T^t is the transpose of matrix T. While T represents a directed tour, U represents an undirected tour, which can be convenient when working with properties of the tour that are independent of direction. Unlike the 3-SAT and Ising spin glass problems, the natural basic move made in order to attempt an improvement of a configuration is not a single spin flip but a 2-opt move [Flood, 1956; Croes, 1958], especially as a single spin flip in U results in an invalid tour. A 2-opt move involves the removal of two edges and the replacement of two different edges into the tour. With respect to using U as a configuration representation, four Boolean variables would need to be changed to accomplish a 2-opt move. This makes the implementation of the TSP more involved than the 3-SAT. If a spin variable $S_{\langle ij \rangle} = 2U_{ij} - 1 = \pm 1$ is defined, then in order to remove edges $(c_1 \rightarrow c_2)$ and $(c_3 \rightarrow c_4)$ and replace them with edges $(c_1 \rightarrow c_3)$ and $(c_2 \rightarrow c_4)$, four spin variables involving the edges concerned need to be flipped. The tour length or Hamiltonian to be minimized can now be expressed as

$$\mathcal{H}_p = \sum_{\langle ij \rangle} d_{ij} \frac{\left(S_{\langle ij \rangle} + 1\right)}{2} \qquad (2.10)$$

In trying to progress to the next stage of converting $\mathcal{H}_p$ into a quantum Hamiltonian with a Γ-dependent term that respects the 2-opt move, the authors of [Martoňák et al., 2004] noticed that it would be difficult to apply a Suzuki-Trotter transformation to the resulting Hamiltonian. It was therefore decided that the same single spin flip system for the 3-SAT be used for the TSP as an approximation. This way, (2.9) can be used as the new classical Hamiltonian, while still performing 2-opt moves during the Metropolis Monte Carlo simulation to maintain feasibility. Their experiments on a 1001-city instance from TSPLIB [Reinelt, 1991] showed that QA successfully outperformed CA. Perhaps, the most important lesson from the study is that when more complicated moves than a single spin flip are required for a particular problem, the system in (2.9) can still be tried as an approximation. As will be seen in chapter 4, this turns out to be invaluable for the graph k-colouring problem.

2.4 Conclusion

The aim of this chapter was to review the Path Integral Monte Carlo formulation for quantum annealing (PIMC-QA). As the Ising spin glass problem is a combinatorial optimization problem belonging to classical and quantum physics, it was natural to start from this problem while presenting the features of quantum annealing. We reviewed characterising a cost function of a problem as a potential energy, expressing this as a quantum Hamiltonian, and then applying the Suzuki-Trotter transformation in order to obtain a new classical Hamiltonian with P replicas that can be simulated feasibly on a classical computer.

With the Boolean satisfiability problem, we moved from an optimization problem originating in physics to a domain more closely associated with computer science. However, similar principles still applied as a Boolean variable can be treated as an Ising spin, with the cost function re-written accordingly. Even though previous applications of PIMC-QA did not show an advantage over classical annealing for Boolean satisfiability, they clearly showed the steps involved in applying PIMC-QA to a given problem. The application of PIMC-QA to the travelling salesman problem was more involved because single spin flips were not sufficient to change one valid configuration to another valid configuration. However, previous authors introduced an approximation that nevertheless resulted in a working algorithm.

Chapter 3

Heuristic Approaches to the Graph Colouring Problem

This chapter reviews the graph colouring problem in the context of heuristic algorithms. Section 3.1 presents the problem description and definitions. Section 3.2 addresses greedy construction algorithms, while Section 3.3 reviews classical simulated annealing algorithms. In Section 3.4, Tabu search algorithms are reviewed. Evolution hybrid algorithms are reviewed in Section 3.5, while Section 3.6 addresses set extraction algorithms.

3.1 Problem Description and Definitions

By the graph colouring problem, we refer to the most common variant involving finding the minimum number of colours required to label the vertices of a graph, so that no adjacent vertices receive the same colour. Let $G = (V, E)$ be an undirected loop-free graph, where V is the set of vertices and E is the set of edges. Given a set of k colours, a proper k-colouring of G is a mapping $\varphi: V \rightarrow \{1,2, \dots, k\}$ such that $\varphi(u) \neq \varphi(v)$, for all $\{u, v\} \in E$. The graph k-colouring problem can be defined as that of deciding whether any proper k-colouring exists for a given instance (G, k). Figure 3.1 shows a graph that has been properly coloured.

The graph colouring problem itself asks for the chromatic number χ, which is the smallest value of k for which a proper k-colouring exists for a given graph G. Clearly, any procedure which solves the k-colouring problem can solve the graph colouring problem by repeated application for successively decreasing values of k.

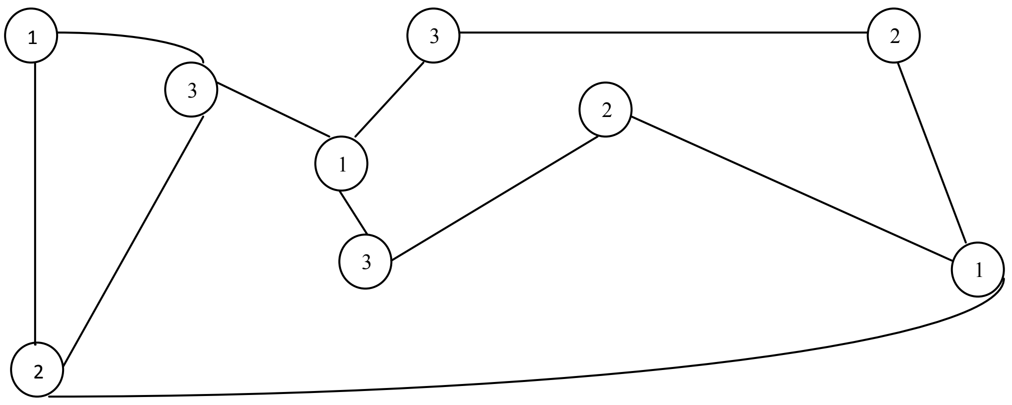

Figure 3.1: A properly coloured graph

There are some known easy special cases including planar graphs, which are always 4-colourable [Appel & Haken, 1977], and 2-colourability, which is always decidable in polynomial time for any graph. However, in general, the k-colouring problem is NP-complete [Karp, 1972; Garey & Johnson, 1979], and thus the graph colouring problem is NP-hard.

Algorithms that attempt to find the chromatic number of a graph by exhaustive enumeration and backtracking search have been developed in the past [Brown, 1972; Brélaz, 1979], and continue to be investigated [San Segundo, 2012]. Due to the difficulty of the problem, exact solvers are still mostly applicable to graphs with less than a hundred vertices. In order to address larger instances, heuristics which do not guarantee that the chromatic number would be found, but nevertheless offer practical advantages, have become indispensable. We review construction based algorithms in Section 3.2; classical simulated annealing algorithms in Section 3.3; Tabu search algorithms in Section 3.4; Evolutionary-Tabu hybrid algorithms in Section 3.5 and independent set extraction pre-processing in Section 3.6.

3.2 Greedy Construction Algorithms

One of the earliest ideas for obtaining approximate solutions to the graph colouring problem (and several other combinatorial optimization problems) is that of a greedy construction algorithm, sometimes also referred to as successive augmentation [Johnson et al., 1991] . A greedy construction algorithm for the graph colouring problem starts with a partial colouring involving only a few vertices, which is then extended until all vertices have been coloured. Notably, unlike a backtracking approach, the colours given to vertices are not usually reconsidered. This usually leads to a fast polynomial-time algorithm that approximates the chromatic number of a graph.

The most basic construction algorithm for graph colouring starts with some permutation $v_1, v_2, \dots, v_n$ of the vertices. Each vertex v_i is then sequentially assigned the numerically smallest colour not already taken by any of its adjacent vertices. After the last vertex v_n has been coloured, the total number of colours used provides an upper bound on the chromatic number χ. This algorithm is known as SEQ in the heuristics literature [Johnson et al., 1991; Malaguti et al., 2008]. SEQ also provides proof that every graph can be properly coloured with $\Delta + 1$ colours, where Δ is the maximum degree of a graph [Diestel, 2000]. The degree of a vertex is the number of vertices adjacent to it. In practice, it is usually possible to improve on $\Delta + 1$ when colouring with SEQ. Multiple executions can also be carried out using a different random permutation each time, after which the best result can be chosen.

Instead of simply colouring the vertices in the sequence of a random permutation, it is possible to use more sophisticated rules to select which vertex should be coloured next, in an attempt to improve on the results of SEQ. This is what the construction version of the maximum degree saturation (DSATUR) algorithm does [Brélaz, 1979]. DSATUR is based on the heuristic idea that fewer colours may be used in the long run if the vertex with the largest chromatic degree is coloured first. The vertex with the largest chromatic degree is one whose adjacent vertices already use the largest number of colours. Experiments with DSATUR have demonstrated its practical advantages over SEQ by providing better upper bounds to the chromatic number of many graphs [Johnson et al., 1991].

A slightly different approach to greedy construction consists of building independent sets one after the other. An independent set refers to a group of vertices in which no two vertices share an edge. All the vertices in an independent set can therefore be assigned the same colour. This approach was originally presented as the recursive largest first (RLF) algorithm [Leighton, 1979]. Experiments from [Johnson et al. 1991] suggest that RLF can outperform DSATUR and SEQ on a wide variety of graphs. Although greedy construction algorithms often offer a quick practical way to obtain reasonable colourings, the results can be very poor in the worst case [Halldórsson, 1993; Zuckerman, 2006]. Additionally, it is difficult to exploit additional running time when available. Stochastic local search approaches attempt to improve on this situation.

3.3 Classical Simulated Annealing Algorithms

Classical simulated annealing (CA) algorithms were among the first stochastic local search algorithms to be attempted on graph colouring [Chams et al., 1987; Johnson et al., 1991; Morgenstern, 1996]. While greedy construction heuristics run quickly but often produce colourings very far from optimal, an algorithm based on CA aims to spend more time in the hope of ending up with a better solution. Central to the application of local search to any problem is the definition of a cost function (or potential energy) to be minimized. Additionally a search space and a neighbourhood function for moving around the search space need to be chosen. Three variants of CA algorithms were presented in [Johnson et al., 1991], one of which was similar to that in [Chams et al., 1987]. A fourth one was investigated in [Morgenstern, 1996]. What follows is a review of all four.

3.3.1 Penalty function

The first CA algorithm considered in [Johnson et al., 1991] was a penalty-function approach. In the penalty-function approach, a configuration in the search space is represented as a partition of V into k disjoint subsets $C_1, C_2, \dots, C_k$, where k can vary between 1 and V inclusively. We note that k could have been defined to be in the range $1 \leq k \leq \Delta + 1$, where Δ is the maximum degree of the graph, as $\Delta + 1$ colours is always sufficient to colour any graph [Diestel, 2000]. The search space explored includes any such partition of V, whether any C_i contains conflicting edges or not. Two configurations are defined to be neighbours if one could be derived from the other by the movement of one vertex from one subset (or colour class) to another, creating a new colour class if necessary. A neighbour is generated by first randomly selecting a vertex v from a randomly selected nonempty colour class C_{OLD}. Next an integer i is randomly chosen in the range $1 \leq i \leq k + 1$, where k is the current number of colours. Finally the neighbour is obtained by moving v to colour class i. The potential energy to be minimized is given as $\mathcal{H}_p = -\sum_{i=1}^{k} |C_i|^2 + \sum_{i=1}^{k} 2|C_i|.|E_i|$, where E_i is the set of conflicting edges contained in C_i.

The main idea behind the term $= -\sum_{i=1}^{k} |C_i|^2$ is to encourage movement to parts of the search space where colour classes are larger, in order to indirectly influence k, which is to be minimized. The other term $\sum_{i=1}^{k} 2|C_i|.|E_i|$ penalizes configurations possessing conflicting edges in their colour classes. The penalty function algorithm represents a case where the cost function itself is not exact, but only a heuristic. While the true global minimum of $\mathcal{H}_p$ does not necessarily correspond to the chromatic number χ, the idea was shown to work reasonably well in practice. An interesting property of $\mathcal{H}_p$ noted in [Johnson et al., 1991] is that every local minimum of $\mathcal{H}_p$ is a proper colouring. With the main components of the penalty function algorithm defined, experiments were carried out in the usual CA fashion by generating new random neighbours, unconditionally accepting those with $\Delta\mathcal{H}_p < 0$, or otherwise accepting the new neighbour with a probability $\exp(-\Delta\mathcal{H}_{\mathrm{p}}/\mathrm{T}_{\mathrm{t}})$, where $\Delta\mathcal{H}_p$ is the change in the cost function and T_t is the temperature at time t.

3.3.2 Kempe chain neighbourhood

The second CA algorithm considered in [Johnson et al., 1991] is the Kempe chain neighbourhood approach, which uses the same potential energy as in the penalty function approach, but only permits proper colourings. The cost function therefore reduces to $\mathcal{H}_p = -\sum_{i=1}^{k}|C_i|^2$. For a given proper coloring, $\{C_1, C_2, \dots, C_k\}$ of graph G, a Kempe chain K can be obtained by selecting any two different colour classes C_i and C_j and locating a maximal connected component in the subgraph of G induced by $C_i \cup C_j$. Provided that $K \neq C_i \cup C_j$, a new neighbouring proper colouring can be derived by replacing C_i with $(C_i \backslash K) \cup (C_j \cap K)$, and C_j with $(C_j \backslash K) \cup (C_i \cap K)$. This forms the basis for a different CA algorithm where $\mathcal{H}_p$ can be minimized in the same fashion as in the penalty function approach. An advantage of this system over the penalty function approach is that the current state of the algorithm always consists of a proper colouring, thus allowing the process to be terminated at anytime for the retrieval of an approximate solution.

3.3.3 Fixed-*k* neighbourhood

The third CA algorithm presented in [Johnson et al., 1991] fixes the value of the number of colours k, and tries to find a proper k-colouring by minimizing the total number of conflicting edges to zero. If this succeeds, the process can then be repeated with $k - 1$, and so on. One of the first uses of this idea can be found in [Chams et al., 1987]. The neighbourhood is defined such that a randomly selected conflicting vertex in one colour class moves to a different randomly selected colour class. It was observed that this asymmetric neighbourhood was more effective than the one in which any vertex was allowed to move to any colour class, as the asymmetric neighbourhood forces the algorithm to deal with the conflicting vertices. Experiments in [Johnson et al., 1991] did not show any of the simulated annealing approaches to be dominant over the others. However, over the years, the fixed-k neighbourhood has turned out to be very effective, whilst not being unnecessarily complicated, and is now used in many of the leading algorithms including those based on Tabu search. Particularly, the operations for the fixed-k neighbourhood are cheaper than those in the Kempe chain neighbourhood. The fixed-k neighbourhood forms the basis of the quantum annealing algorithms for graph colouring designed and studied in chapters 4, 5 and 6 of this thesis.

3.3.4 Impasse-class neighbourhood

An alternative to the standard fixed-k neighbourhood was presented in [Morgenstern, 1996] and termed the impasse-class neighbourhood. The idea is that although k is still fixed, a proper colouring covering a subset of the vertex set is always maintained, and all vertices that cannot yet legally occupy any of the k colour classes are placed together in an impasse class. Emptying this impasse class now becomes the objective. The basic local search move consists of selecting a vertex v at random from the impasse class, selecting a colour class i at random to place this vertex, and moving any vertex adjacent to v in i into the impasse class to maintain the feasibility of the partial colouration. Although the cost function to be minimized in this system can simply be taken as the size of the impasse class, the sum of the degrees of the vertices in the impasse class was used instead. This was done in order to encourage vertices with lower degrees to remain in the impasse class the longest, thereby dealing with the potentially more difficult vertices having higher degrees first. Classical simulated annealing with a fixed temperature (also known as the Metropolis algorithm) was used to search the impasse-class neighbourhood. Experiments in [Morgenstern, 1996] showed the impasse-class neighbourhood to be a strong approach to metaheuristic graph colouring. It was combined with additional neighbourhoods and pre-processing techniques to produce results that improve upon those of [Johnson et al. 1991].

3.4 Tabu Search Algorithms

Tabu search [Glover, 1989; Glover, 1990] is a successful alternative to simulated annealing designed to be more reliant on learning than on the use of a random number generator. The signature component of Tabu search is a Tabu list which stores attributes of the configurations of the search space already visited. By using the Tabu list to avoid recently visited configurations, Tabu search aims to prevent cycling and getting trapped by poor local minima. One of the earliest implementations of Tabu search for graph colouring was in [Hertz & Werra, 1987]. It uses the same fixed-k neighbourhood discussed in Section 3.3.3. An initial configuration is improved in several iterations, each of which consists of the random sampling of a given amount of configurations from the neighbourhood, and making the best move not prevented by the Tabu list. Every time a move is made, the reverse move is added to a Tabu list which was defined to be of fixed size and of length seven. The reverse move serves as an abbreviated form of the configuration that is prohibited in the short term. As it is the moves that are stored in the Tabu list rather than the prohibited configurations themselves, there is a risk that certain high-quality configurations that had actually never been visited are disallowed as well. To alleviate this problem, Tabu search usually employs the aspiration criterion that if a configuration is better than all others visited before, its Tabu status is overridden. It was noted in [Hertz & Werra, 1987] that the Tabu search outperformed a contemporaneous simulated annealing algorithm [Chams et al., 1987] developed by the same group of authors.

The original Tabu search implementation of [Hertz & Werra, 1987] was subsequently improved in several other studies. With the advent of increased computing power, it became more feasible to search for the best move in the whole of the immediate neighbourhood, rather than just a random sample of it [Fleurent & Ferland, 1996]. It was also observed that randomly varying the length for the Tabu list produced a more effective algorithm [Fleurent & Ferland, 1996; Dorne & Hao, 1998a; Blöchliger & Zuffery, 2008] than the static alternative in [Hertz & Werra, 1987]. Enhancements to the standard cost function of simply counting the number of conflicting edges have been investigated for the fixed-k neighbourhood [Porumbel et al., 2008]. The standard cost function was modified to bias the search in such a way that conflicts involving vertices with higher degrees are likely to be resolved first, much like it was done for simulated annealing in [Morgenstern, 1996]. Additionally, terms based on the move frequency history of the search were incorporated into the standard cost function. A Tabu search algorithm based on the impasse-class neighbourhood discussed in Section 3.3.2 was presented in [Blöchliger & Zuffery, 2008]. Tabu search is often used as the local search component of Evolutionary hybrid algorithms which are reviewed in Section 3.5.

3.5 Evolutionary Hybrid Algorithms

Evolutionary algorithms [Bäck, 1996] maintain a population of individuals, which improve by competing and co-operating with each other via operators such as mutations, crossovers, and selection processes over several generations. Instead of maintaining a large population in which each individual undergoes mutations, it was recognized that a much smaller population could evolve effectively with each individual performing a deep local search during each generation. This paradigm is referred to as an evolutionary hybrid or a memetic algorithm [Moscato, 1989; Moscato, 2003; Moscato 2005].

Early studies of evolutionary hybrid algorithms in the context of graph colouring were carried out in [Fleurent & Ferland, 1996] and [Galinier & Hao, 1999]. Recombination in [Fleurent & Ferland, 1996] used a string-based representation for the colouring configuration, so that traditional genetic operators such as the 1-point, 2-point and uniform crossovers could be used. In contrast, [Galinier & Hao, 1999] used specialized crossover operators that emphasized graph colouring as a grouping problem [Falkenauer, 1994; Falkenauer, 1998], and built each offspring by blending colour classes from two individuals. As a result, it outperformed the approach in [Fleurent & Ferland, 1996] and became one of the most powerful algorithms of its time. The crossover in [Galinier & Hao, 1999] was named the Greedy Partition Crossover (GPX). GPX built a new offspring by alternating between two parents and copying the largest colour classes. Extensions of the GPX were later studied by [Lü & Hao, 2010; Porumbel et al., 2010b] in the context of multi-parent recombination [Eiben & Bäck, 1997] operating on up to six parents at once.

The algorithms in [Lü & Hao, 2010; Porumbel et al., 2010b] also featured improved population diversity control and selection with the use of the partition distance [Gusfield, 2002] between individuals. Another algorithm using similar ideas but with the impasse-class neighbourhood was developed in [Malaguti et al., 2008].

3.6 Independent Set Extraction Pre-processing

Before applying any of the colouring algorithms already discussed to any graph, there is the possibility of first looking for large independent sets and removing them in order to colour a smaller residual graph. This idea was used early on in many graph colouring studies [Bollobás & Thomason, 1985; Chams et al., 1987; Hertz & Werra, 1987; Fleurent & Ferland, 1996; Morgenstern, 1996]. The resulting upper bound to the chromatic number is the number of independent sets extracted, plus the number of colours required by the residual graph.

Independent set extraction pre-processing has been mostly effective for medium density random graphs. After hybrid evolutionary algorithms began to produce very good results on their own, set extraction fell out of use. It was later revived with a more powerful version than had ever been devised [Wu & Hao, 2012].

Previous versions iteratively sought a largest possible independent set, immediately removing each one as soon it was found. If there were multiple independent sets of the same size, then refinements mainly consisted of selecting one covering the most edges, and removing it immediately. However, in [Wu & Hao, 2012], whenever an independent set of size z is found (with a Tabu search heuristic), instead of extracting this immediately, additional independent sets of size z are sought and placed into a bank B. Clearly the independent sets in B would usually not be mutually exclusive. A Tabu search heuristic then approximately solves an NP-hard set packing problem to find a maximum number of mutually exclusive sets in B, which are then extracted all at once from the graph. The procedure is then repeated until a suitably sized residual graph is left, which is then coloured with more conventional methods. This way of extracting independent sets allows more vertices to be covered than was previously possible with earlier versions for the same number of independent sets, thus leading to better results on some types of graphs.

3.6 Conclusion

The aim of this chapter was to review the graph colouring problem from the perspective of heuristic algorithms. While greedy construction algorithms terminate very quickly in polynomial time, they often provide very weak upper bounds for the graph colouring problem. Attempts to improve on the results of greedy construction while making use of additional running time involves the use of algorithms based on local search such as classical simulated annealing, Tabu search and Evolutionary hybrid algorithms which were all addressed. The population based algorithms are known to be more powerful than local search approaches maintaining only a single configuration.

Chapter 4

A Quantum Annealing Algorithm for the Graph Colouring Problem

This chapter is concerned with the design of a competitive graph colouring algorithm based on quantum annealing. The graph colouring problem can be approached heuristically by solving a series of k-colouring problems for progressively lower values of k, thereby obtaining an upper bound for the chromatic number. Given an undirected graph $G = (V, E)$ with $|V| = n$, G is k-colourable if there exists a mapping $\varphi: V \to \{1,2, \dots, k\}$ such that $\varphi(i) \neq \varphi(j)\ \forall\{i,j\} \in E$.

An arbitrary assignment of vertices to colours can be denoted by the configuration $\omega(\varphi(1), \varphi(2), \dots, \varphi(n))$. An alternative representation of a configuration ω is as a partition of the vertex set V into k subsets $V_1, V_2, \dots, V_k$, where vertices given the same label belong to the same subset. Depending on the configuration, some of the subsets may be empty. The partition-based representation is known to be important in the design of population-based metaheuristics for the graph colouring problem, as it can alleviate symmetry problems resulting from the redundancy of colour naming [Galinier & Hao, 1999].

In order to cast k-colouring as a combinatorial optimization problem, we require a cost function. Given that a solution must not contain any conflicting edges, a natural choice is to minimize the number of conflicts. The set Ω of all possible assignments of vertices to the available k colours is the search space from which we seek a solution. A solution is any configuration $\omega \in \Omega$ such that $\forall\{i,j\} \in E$, $\varphi(i) \neq \varphi(j)$.

We denote the cost function representing the number of conflicting edges by $\mathcal{H}_p$. A graph is k-colourable if and only if there exists at least one configuration ω such that $\mathcal{H}_p(\omega) = 0$. Any such ω is a global minimum of the cost function $\mathcal{H}_p$. This approach is the fixed-k formulation discussed in Section 3.3.3. Its main strength is that a potential energy of zero provides evidence that a global minimum has been reached. Furthermore, there is nothing to suggest that the fixed-k formulation is any worse than the alternatives such as the kempe chain and penalty approaches discussed in Section 3.3. For example, the experiments in [Johnson et al., 1989] demonstrate that no approach dominates the others. Additionally, the current leading algorithms use the fixed-k approach [Lü & Hao, 2010; Porumbel et al., 2010b].

A basic local search approach can start by initializing ω uniformly at random. Afterwards, a conflicting vertex is selected at random to have its colour changed to a new randomly chosen one, giving a new configuration ω'. If $\mathcal{H}_p(\omega')$ is less than $\mathcal{H}_p(\omega)$, then ω' is accepted. Otherwise another ω' is generated and tested. After some iterations of generating, testing and moving to configurations with a lower value for the cost function, a local minimum will eventually be reached. Only for the simplest of problem instances would such a local minimum also be a global minimum. In a typical problem instance, there are usually an exponential number of local minima which trap the process described. However, the basic local search serves as a fundamental principle for more sophisticated local search approaches.

One improvement over the basic local search is the well-known approach of Classical (simulated) Annealing (CA) [Kirkpatrick et al., 1983; Cerný, 1985], in which the cost function is regarded as a potential energy. Instead of only accepting downward movements in the potential energy landscape, upward movements are also eligible according to the

Metropolis criterion. Thermal fluctuations provide an opportunity of escape from poor local minima, and increase the chance that a global minimum will eventually be found.

The main idea of Quantum Annealing (QA) [Santoro et al., 2002; Martonák et al., 2004; Das & Chakrabarti, 2008] is to introduce quantum fluctuations to perform an analogous role to thermal fluctuations of facilitating the escape from poor local minima. Quantum fluctuations allow tunnelling through barriers rather than having to scale the height of these barriers thermally. In addition to the potential energy $\mathcal{H}_p$, we need to provide a kinetic energy that will provide the quantum fluctuations. This is usually done by expressing the combinatorial optimization problem at hand in terms of an Ising spin system, and then defining the kinetic energy as some intuitive property of the spins [Martonák et al., 2004; Battaglia et al. 2005].

In Section 4.1, we define a suitable spin system and present a sketch of our quantum annealing algorithm for graph k-colouring. Section 4.2 deals with the provision of efficient procedures and essential data structures for critical components of the k-colouring quantum annealing algorithm. Implementation specific considerations such as the random number generator and code optimization techniques are addressed in Section 4.3. In Section 4.4, the results of experiments in which quantum annealing is compared to other algorithms are reported. Quantum annealing is found to be competitive. Particularly, our algorithm became the first to report a 222-colouring for the DIMACS competition graph DSJC1000.9. This was published in a journal article [Titiloye & Crispin, 2011a]. This result was subsequently reported independently by [Wu & Hao, 2012] with the aid of set extraction pre-processing and an Evolutionary-Tabu algorithm.

4.1 A Quantum Annealing Formulation

4.1.1 The Suzuki-Trotter transformation of a k-colouring Ising model

An Ising model consists of spins (or spin variables) which can only be in one of two states [Cipra, 1987]. The standard values of these spins are either up $(+1)$ or down (-1). The k-colouring problem can be defined in terms of Boolean variables, and hence Ising spin variables. If there are several ways to do this, then it is important to choose one that reflects the problem-specific issues of the combinatorial optimization at hand [Martoňák et al., 2004]. Possibly the most straightforward idea is to a have one Boolean variable y_{ik} for every vertex-colour combination to denote whether the combination is present in the current configuration. In this system, there would be a total of $|V|.k$ variables, and any configuration ω could be represented by setting the Boolean variables to appropriate values. An undesirable feature of this system is that configurations that are essentially the same except for colour labelling do not correspond to the same set of Boolean variables. This is the problem of symmetry. As discussed in chapter 2, a Suzuki-Trotter transformation produces a population of interacting replicas. In the case of graph colouring, if the interacting replicas do not respect the highly symmetric nature of the problem, the effectiveness of the interaction will degrade. This has been recognized in past studies of population-based algorithms for the graph colouring. For example, although both are Evolutionary-Tabu hybrid graph colouring algorithms, the approach in [Galinier & Hao, 1999] outperformed that of [Fleurent & Ferland, 1996] by providing crossover operators that handled symmetry better. Even graph colouring algorithms that are not population based benefit from dealing with symmetry properly [Ramani et al., 2004; Méndez-Díaz & Zabala, 2008].

We therefore introduce an alternative system in which each Boolean variable is concerned with whether one vertex has the same colour as another vertex, and not about the specific colour of any vertex. In this system, there is one Boolean variable x_{ij} for every unique pair of vertices. Specifically, $x_{ij} = 0$ if $\varphi(i) \neq \varphi(j)$, and $x_{ij} = 1$ otherwise. Every configuration ω can be expressed as a set $\{x_{ij}\}$ of N constrained Boolean variables, where $1 \leq i \leq n$ and $1 \leq j \leq n$, and $i > j$ for the uniqueness of vertex pairs. The total number of Boolean variables N is the same as the number of edges in a complete graph of n vertices, and is therefore equal to $\binom{n}{2}$ or $n(n-1)/2$. The variables $\{x_{ij}\}$ treat a colouring as a partition of the vertex set, rather than simply an assignment of vertices to any specific colours, thereby capturing a problem-specific essence of graph colouring. As a result, configurations that differ only because of a permutation of colours are nevertheless represented by the same set of Boolean variables. When the k-colouring problem is viewed as an assignment of vertices to colours, there are a total of k^n possible configurations, some of which are the same but for colour naming. Not all of the 2^N possible states of $\{x_{ij}\}$ correspond to valid members of the k-colouring search space. Instead, the total number of valid states of $\{x_{ij}\}$ is the same as the number of possible partitions of the vertex set V into up to k subsets. While this should be less than k^n, there is no simple expression for it. The total number of possible settings of $\{x_{ij}\}$ is given as $\sum_{i=1}^{k} \mathcal{S}(n,i)$, where $\mathcal{S}(n,k)$ denotes Sterling numbers of the second kind [Stanley, 2011]. The quantity $\mathcal{S}(n,k)$ is the number of ways in which a set containing n elements can be partitioned into exactly k parts, and is defined by the recurrence relation $\mathcal{S}(n,k) = k\mathcal{S}(n-1,k) + \mathcal{S}(n-1,k-1)$. An Ising model can be obtained from the Boolean variables $\{x_{ij}\}$ by defining a set of spins $\{S_{ij}\}$ where $S_{ij} = 1 - 2x_{ij} = \pm 1$. The potential energy represented by the number of conflicting edges in terms of the spins is given as

$$\mathcal{H}_p(\{S_{ij}\}) = \sum_{ij \in E} (1 - S_{ij})/2 \tag{4.1}$$

The Hamiltonian $\mathcal{H}_p$ in (4.1) can already be used in a CA algorithm as $\mathcal{H}_p$ correctly counts the number of conflicting edges in a configuration. Only valid configurations of the k-colouring search space should be sampled however. Therefore, instead of generating a new configuration by a single spin flip with the likelihood of ending up with an invalid one, validity is guaranteed by making multiple spin flips that correspond to the movement of a vertex from one colour class to another. Just as in the case of the Ising Model in Section 2.1 in chapter 2, the Hamiltonian in (4.1) can be converted to a quantum Hamiltonian with a transverse field to yield:

$$\mathcal{H}_q = \mathcal{H}_p(\{\sigma_{ij}^z\}) - \Gamma \sum_{ij} \sigma_{ij}^x \tag{4.2}$$

The terms σ_{ij}^z and σ_{ij}^x are Pauli matrices. The Γ-dependent term in (4.2) from a standard Ising model implies that a single spin flip in a valid configuration can always produce another valid configuration, which is true for the standard Ising Model [Martoňák et al., 2002], but not for our *k*-colouring Ising Model, which has its spins constrained to one another in the sense that a single spin flip is in general not enough to move from one valid configuration to another valid one. The same problem was faced in the travelling salesman problem, where it was observed that applying a Suzuki-Trotter transformation to a quantum Hamiltonian with multiple spin flip operators is complicated, whereas the standard model with a single spin flip operator is trivially dealt with [Martoňák et al., 2004]. As a result of this, the travelling salesman study used the equivalent of (4.2) with a single spin flip operator as an approximate formulation.

We follow the same approach here. Just as in the case of the Ising Model in Section 2.1 in chapter 2, applying the Suzuki-Trotter transformation on the quantum Hamiltonian in (4.2) yields a standard classical Hamiltonian

$$\mathcal{H} = \frac{1}{P}\sum_{\rho=1}^{P} \mathcal{H}_p(\{S_{ij,\rho}\}) - J_\Gamma \sum_{\rho=1}^{P}\sum_{ij} S_{ij,\rho}S_{ij,\rho+1} \tag{4.3}$$

The classical Hamiltonian $\mathcal{H}$ in (4.3) can be viewed as consisting of P replicas of the original classical Hamiltonian $\mathcal{H}_p$ in (4.1), with the Γ-dependent term serving as the kinetic energy. The spin $S_{ij,\rho}$ represents the ijth spin of the ρth replica, while J_Γ is given as:

$$J_\Gamma = -\frac{T}{2}\ln\tanh(\Gamma/PT) > 0 \tag{4.4}$$

In order for the classical Hamiltonian in (4.3) to be an exact equivalent of the quantum one in (4.2), P has to tend to tend to infinity. Obviously this cannot be done in a practical Monte Carlo simulation, and a small value of P has to be used. We connect a finite and small number of the replicas in a ring, and revise (4.3) as:

$$\mathcal{H} = \frac{1}{P}\sum_{\rho=1}^{P} \mathcal{H}_p(\{S_{ij,\rho}\}) - J_\Gamma \left(\sum_{\rho=1}^{P-1}\sum_{ij} S_{ij,\rho}S_{ij,\rho+1} + \sum_{ij} S_{ij,1}S_{ij,P}\right) \tag{4.5}$$

The reason for connecting them in a ring is so that each replica is identical to the others in terms of sharing spin products with exactly two other replicas. This helps simplify the algorithm as the first and last replicas do not have to be taken as special cases. For computational purposes we take $\mathcal{H}$ as the expression in (4.5).

4.1.2 CA-Col: A classical annealing algorithm for the k-colouring problem

Before presenting the quantum annealing algorithm for the graph k-colouring, it is helpful to review the CA approach, due to the relationship between them. A classical simulated annealing graph colouring algorithm called CA-Col is presented in Algorithm 4.1. The cost function used counts the number of conflicting edges, and is therefore the same as $\mathcal{H}_p$. One configuration is obtained from another by changing the colour of a conflicting vertex. In other words, the neighbourhood of a configuration is the set of all configurations that can be reached from the current one by choosing a new colour for a conflicting vertex.

Algorithm 4.1: Classical annealing for the k-colouring problem

```
1.  Input: $G$, $k$, $M$, $T_0$, $MaxSteps$, and estimate neighbourhood size $N_S$ as $|V|.k$
2.  Output: "Yes" if proper $k$-colouring is found or "No" otherwise
3.  Randomly initialize a configuration $\omega$, and initialize $T \leftarrow T_0$
4.  While termination condition is not met
5.    $attempted \leftarrow 0$
6.    While $attempted < M.N_s$
7.      Randomly select a vertex $v$ conflicting in $\omega$
8.      Change colour of $v$ to a new randomly selected colour to derive $\omega'$
9.      $\Delta\mathcal{H}_p \leftarrow \mathcal{H}_p(\omega') - \mathcal{H}_p(\omega)$
10.     If $\Delta\mathcal{H}_p < 0$
11.       $\omega \leftarrow \omega'$
12.     Else
13.       With probability $\exp(-\Delta\mathcal{H}_p/T)$, $\omega \leftarrow \omega'$
14.     $attempted \leftarrow attempted + 1$
15.   End While
16.   $T \leftarrow T - (T_0/MaxSteps)$
17. End While
```

Algorithm 4.1 is essentially the same as the classical annealing algorithm for the k-colouring problem presented in [Chams et al., 1987; Johnson et al., 1991]. On line 1, we input the graph G, the number of available colours k, and starting temperature T_0. We also input an integer multiplier M for the neighbourhood size, and a maximum number of Monte Carlo steps $MaxSteps$. The neighbourhood size N_s is estimated as $|V|.k$. Line 2 describes the required output, which is either an acknowledgement of success with evidence, or failure of the algorithm.

As classical annealing belongs to a class of algorithms termed incomplete, it is unable to confirm that a k-colouring absolutely does not exist. Line 3 chooses one of the k colours for each vertex uniformly at random to provide an initial configuration ω. The temperature is also initialized. On line 4, a loop is started, which ends when a termination condition is met. A termination condition could be a time limit, or when the temperature has been decreased to zero. The algorithm is definitely stopped if a solution has been found, though this is not explicitly shown in Algorithm 4.1 to simplify the flow of the algorithm. We start a counter of attempted moves on line 5 before starting an inner loop on line 6, which operates until the number of attempted moves has reached $M.N_s$.

On line 7, a vertex v is selected uniformly at random from the set of conflicting vertices and assigned a new randomly selected colour on line 8 to yield a new configuration ω'. The change in the value of the cost function is computed on line 9 as $\Delta\mathcal{H}_p$. If the change in cost function represents a decrease on line 10, then ω' is accepted as the new ω on line 11. Otherwise we proceed to line 13, where a random number $r \in [0,1)$ is generated and compared to $\exp(-\Delta\mathcal{H}_p/T)$. If $\exp(-\Delta\mathcal{H}_p/T)$ is greater than r, then ω' is accepted. This is known as the Metropolis criterion. Line 15 ends the inner loop and a Monte Carlo step. On line 16, T is decreased according to a linear schedule [Strenski & Kirkpatrick, 1991].

4.1.3 QA-Col: A quantum annealing algorithm for the k-colouring problem

Algorithm 4.2 describes our quantum annealing algorithm for the k-colouring problem. While classical annealing maintains a single configuration and attempts to minimize the number of conflicting edges, quantum annealing maintains P interacting replicas that attempt to minimize the shared and more complex Hamiltonian $\mathcal{H}$ in (4.5). Line 1 depicts the input of the graph G, available colours k, multiplier M, initial temperature T_0 and $MaxSteps$, all present in Algorithm 4.1 for CA as well. In addition, we input the number of replicas P, and the initial value of the field strength parameter Γ_0.

Algorithm 4.2 Quantum annealing for the k-colouring problem

```
1.  Input: $G$, $k$, $M$,$P$, $T_0$,$\Gamma_0$, $MaxSteps$
2.  Initialize: $N_s \leftarrow |V|.k$, $\Gamma \leftarrow \Gamma_0$, $T \leftarrow T_0$
3.  Output: "Yes" if a proper $k$-colouring is found or "No" otherwise
4.  Randomly initialize $P$ configurations $\{\omega_\rho\} \equiv \varpi$
5.  While termination condition is not met
6.    For $\rho = 1, \dots, P$
7.      $attempted \leftarrow 0$
8.      While $attempted < M.N_s$
9.        Randomly select a vertex $v$ conflicting in $\omega_\rho$
10.       Change colour of $v$ to a new randomly selected colour, to derive $\omega_\rho'$ and $\varpi'$
11.       $\Delta\mathcal{H}_p \leftarrow \mathcal{H}_p(\omega_\rho') - \mathcal{H}_p(\omega_\rho)$
12.       $\Delta\mathcal{H} \leftarrow \mathcal{H}(\varpi') - \mathcal{H}(\varpi)$
13.       If $\Delta\mathcal{H}_p < 0$ Or $\Delta\mathcal{H} < 0$
14.         $\varpi \leftarrow \varpi'$
15.       Else
16.         With probability $\exp(-\Delta\mathcal{H}/T)$, $\varpi \leftarrow \varpi'$
17.       $attempted \leftarrow attempted + 1$
18.     End While
19.   End For
20.   $\Gamma \leftarrow \Gamma - (\Gamma_0/MaxSteps)$
21. End While
```

On line 2, the estimation of the neighbourhood size, the temperature, and the field strength are all initialized. In contrast to CA-Col, the temperature is kept fixed throughout, and it is the field strength that is periodically reduced linearly towards zero, according to the maximum number of steps specified, just as in [Santoro et al., 2002; Martoňák et al., 2002]. On line 4, a set of P independent configurations indicated by $\{\omega_\rho\}$ or ϖ is initialized randomly. The structure of the algorithm is such that line 5 starts an outermost loop, which continues until a termination condition is satisfied, while an inner loop gives control to the replicas one at a time on line 6. When each replica executes, it enters an innermost loop on line 8.

On line 9, a conflicting vertex v is randomly selected in the ρth replica, and has its colour changed to a new randomly chosen colour to derive $\omega_\rho{}'$ and hence ϖ' on line 10. On lines 11 and 12, we compute $\Delta\mathcal{H}_p$ and $\Delta\mathcal{H}$ respectively.

Line 13 checks whether either $\Delta\mathcal{H}_p$ or $\Delta\mathcal{H}$ is less than zero, and accepts the change if this is the case. When $\Delta\mathcal{H}_p$ is less than zero, this means a new configuration with less potential energy has been identified. It is accepted as is customary in annealing. Furthermore $\Delta\mathcal{H}_p < 0$, almost always implies that $\Delta\mathcal{H} < 0$ provided that reasonable parameters have been chosen from T and Γ so that $\mathcal{H}_p$ (the actual cost function that we care about) carries a lot of weight. Additionally, the device on line 13 ensures that opportunities to move to a global minimum from configurations where $\mathcal{H}_p = 1$ are not missed. On line 16, the crux of the algorithm, where the Metropolis criterion is tested, it is $\Delta\mathcal{H}$ that is used, rather than $\Delta\mathcal{H}_p$. On line 20, the field strength Γ is decreased linearly [Santoro et al., 2002; Martonák et al., 2004]. The practicality and success of the quantum annealing algorithm depends on how efficiently $\Delta\mathcal{H}$ can be calculated, given how involved $\mathcal{H}$ is, as defined in (4.5). Section 4.2 is devoted to finding incremental techniques to enable a reasonably efficient computation of $\Delta\mathcal{H}$.

4.2 Efficient Procedures and Data Structures for Critical Components of QA-Col

4.2.1 Efficient computation of the potential energy change

Computing the potential energy change $\Delta\mathcal{H}_p$ is an important part of calculating the overall $\Delta\mathcal{H}$. Even though $\mathcal{H}_p$ is expressed in terms of Boolean variables in (4.1), the Boolean variable themselves are not needed in order to compute $\mathcal{H}_p$ or $\Delta\mathcal{H}_p$ quickly, as $\mathcal{H}_p$ is the number of conflicting edges in the current configuration. If $CE(\omega)$ is the number of conflicting edges in ω, then $\Delta\mathcal{H}_p$ can be expressed as $CE(\omega') - CE(\omega)$. Let the set of vertices adjacent to a vertex v in G be denoted by $Adj(v)$. If we propose to move v from colour class V_α to colour class V_β, then $\Delta\mathcal{H}_p$ is equivalent to $|Adj(v) \cap V_\beta| - |Adj(v) \cap V_\alpha|$. This is because edge conflicts with the adjacent vertices of v having colour α would be resolved, while new edge conflicts would be formed with any adjacent vertices of v having colour β. If an array F with dimensions $|V| \times k$ is initialized and maintained such that for any vertex v and any colour θ, $F(v,\theta) = |Adj(v) \cap V_\theta|$, then $\Delta\mathcal{H}_p = F(v,\beta) - F(v,\alpha)$ can be computed in constant time or $O(1)$. QA-Col needs to maintain P separate instances of F, while CA-Col only needs one. The array F is initialized at the start of the algorithm, and only needs to be updated anytime a vertex actually changes colour. In this case, the array F can be updated in $O(V)$ by decrementing $F(u,\alpha)$ by one and incrementing $F(u,\beta)$ by one for all $u \in Adj(v)$. Similar approaches are widely used to compute the cost function while trying to find the best move in Tabu search algorithms for the k-colouring problem [Fleurent & Ferland, 1996; Galinier & Hertz, 2006]. In any annealing algorithm with a high move acceptance ratio, having to update F in $O(V)$ every time a move is accepted is likely to become too expensive. In the CA algorithm for the k-colouring problem in [Johnson et al., 1991], 50-60% of the randomly proposed moves were accepted at the start of the algorithm. This is probably why the incremental approach to calculating $\Delta\mathcal{H}_p$ does not appear to have being used. There is no mention of the approach in [Chams et al., 1987] either. However, quantum annealing aims to maintain a low temperature, while driving the simulation mainly by the field strength. As a result, much lower acceptance ratios than in [Johnson et al., 1991] are expected in QA-Col, making the incremental approach attractive. Additionally, it has been demonstrated that even classical annealing with low starting temperatures and acceptance ratios can be successful for graph colouring [Morgenstern, 1996]. Therefore we used low starting acceptance ratios for our CA experiments as well.

The Hamiltonian $\mathcal{H}$ for quantum annealing in (4.5) can be rewritten as $PEterm - KEterm$, where $PEterm$ is the part containing the potential energy term $\mathcal{H}_p$ and $KEterm$ is the Γ-dependent kinetic energy part. It follows that $\Delta\mathcal{H} = \Delta PEterm - \Delta KEterm$. The change in the potential energy term, $\Delta PEterm,$ can be computed as $\Delta\mathcal{H}_p(\{S_{ij,\rho}\})/P$, where ρ is the replica under consideration. This is because only the currently changing replica enters into the calculation of $\Delta PEterm$. Computing $\Delta KEterm$ reasonably efficiently is more involved and the next section is devoted to this.

4.2.2 Errorless function swindle for the kinetic energy change

The efficient computation of the change in $KEterm$, the Γ-dependent term in the Hamiltonian $\mathcal{H}$ in (4.5) is crucial for the success of quantum annealing. Calculating $\Delta KEterm$ is complicated due to the fact that it involves tracking the change in the products of spins between different replicas. Additionally, because of the definition of the Boolean variables the spins are based on and the constraints between them, several spins usually change in the current replica whenever a single vertex changes colour. An incremental approach is needed, as calculating $KEterm$ from scratch each time would be prohibitively expensive. Each replica ρ is involved in spin products with two other replicas we label b and f, because the replicas are connected in a ring as implied by the Hamiltonian in (4.5). If α and β are the old and new colours respectively, then only the spins of the vertices associated with colour classes V_α and V_β of the current replica ρ can change whenever a move is made in that replica. We define an integer I_KEterm where $KEterm = J_\Gamma . I\_KEterm$, which gives $\Delta KEterm = J_\Gamma . \Delta I\_KEterm$. Recalling the definition of the spin variables in Section 4.1.1, $S_{ij,\rho}$ is $+1$ if $\varphi(i) \neq \varphi(j)$ in replica ρ, and -1 otherwise, where $\varphi(i)$ is the colour of vertex i. If a vertex j is moving from colour class V_α to colour class V_β, then ΔI_KEterm can be computed in $O(V_\alpha + V_\beta)$ as:

$$\Delta I\_KEterm = \sum_{i \in V_\alpha - \{j\}} 2(S_{ij,b} + S_{ij,f}) - \sum_{i \in V_\beta} 2(S_{ij,b} + S_{ij,f}) \tag{4.6}$$

The expression for ΔI_KEterm in (4.6) is derived by noting that any spin products that contain changed spins need to be subtracted and replaced with the products having the updated spin values. Specifically, whenever a vertex j moves from colour class V_α to V_β in the ρth replica, $\forall i \in V_\alpha - \{j\}$, $S_{ij,\rho}$ changes from -1 to $+1$, and $\forall i \in V_\beta$, $S_{ij,\rho}$ changes from $+1$ to -1. As the corresponding spins in the two connected replicas b and f are regarded as staying the same during this move, $\forall i \in V_\alpha - \{j\}$, the quantity $(+1)S_{ij,b} - (-1)S_{ij,b} + (+1)S_{ij,f} - (-1)S_{ij,f}$ which simplifies to $2(S_{ij,b} + S_{ij,f})$ gets contributed to ΔI_KEterm. Similarly, $\forall i \in V_\beta$, the quantity $-2(S_{ij,b} + S_{ij,f})$ gets contributed to ΔI_KEterm. In order to be able to compute the resulting expression in (4.6) in $O(V_\alpha + V_\beta)$, the right data structures have to be used. A list of dynamic disjoint sets is needed to hold the colour classes, and it is desirable to be able to iterate through a colour class in linear time in the size of the given colour class. Since the sum of all the sizes of the sets has to be equal to $|V|$, such a data structure can be designed using perfect hashing with arrays. Furthermore, addition, removal and membership checking can be realised in constant time for the colour classes. Additionally we keep an array data structure B, which stores the colours of the vertices such that $\forall v$, $B(v) = \varphi(v)$. There is no need to store the spins directly in any data structure, as they can be deduced by comparing the colours of the vertices under consideration by using B.

While computing the change in the kinetic energy incrementally in linear time with (4.6) is much more efficient than computing it in quadratic time using (4.5), the computational cost is still high, considering that trillions of evaluations could be needed for large problem instances. Given that an average of $2 * |V|/k$ vertices needs to be checked in order to compute ΔI_KEterm, it can be expensive to iterate through the colour classes of sparser graphs, as they can usually admit smaller values of k than denser graphs with the same number of vertices. However, by first computing an upper bound on ΔI_KEterm, we are able to demonstrate that it is possible to drastically reduce the number of times the expression for which ΔI_KEterm in (4.6) has to be fully computed. Since the expression $\Delta\mathcal{H} = \Delta PEterm - \Delta KEterm$ is used to evaluate move acceptance by comparing $\exp(-\Delta\mathcal{H}/T)$ to a random number $r \in [0,1)$, a smaller value for $\Delta\mathcal{H}$ increases the chance of acceptance. Because J_Γ from (4.4) is always positive, $\Delta\mathcal{H}$ is at its smallest when ΔI_KEterm is at its largest. If a vertex v moves from colour class V_α to V_β then the upper bound on ΔI_KEterm is given as:

$$UB\Delta I\_KEterm = 4(|V_\alpha| + |V_\beta| - 1) \tag{4.7}$$

The expression for $UB\Delta I\_KEterm$ in (4.7) is obtained from (4.6) by substituting $+1$ for all the spins related to V_α, and -1 for all the spins relating to V_β. An inspection of (4.6) shows that it is not possible to obtain a larger value for ΔI_KEterm than $4(V_\alpha + V_\beta - 1)$. Whenever we need to compare $\exp(-\Delta\mathcal{H}/T)$ to $r \in [0,1)$, we first compute a lower bound for $\Delta\mathcal{H}$ as:

$$LB\Delta\mathcal{H} = \Delta PEterm - J_\Gamma . UB\Delta I\_KEterm \tag{4.8}$$

The point is that if a proposed move causes $\exp(-LB\Delta\mathcal{H}/T)$ to be less than r, then we can reject the move, safe in the knowledge that it would have been rejected even if the actual value for $\Delta\mathcal{H}$ had been computed and compared to the same r. On the other hand, if $\exp(-LB\Delta\mathcal{H}/T)$ turns out to be larger than r, only then is it necessary to fully compute the actual $\Delta\mathcal{H}$, in order to compare it with the same r and find out if the move really should be accepted.

Our experiments in Section 4.4.4 show that depending on the problem instance, $LB\Delta\mathcal{H}$ in (4.8) can be used to reject up to 99% of all attempted moves. This is remarkable, considering that the behaviour of the quantum annealing algorithm is not changed in any way apart from its speed. The role of surrogate cost functions in classical annealing was investigated in [Tovey, 1988]. One of the techniques introduced by that author was termed the "surrogate function swindle", with the idea that the annealing algorithm selectively uses an approximate cost function according to an adjustable probability selected to reduce the effect of the errors. The technique of first using $LB\Delta\mathcal{H}$, and only computing the actual $\Delta\mathcal{H}$ in the few cases where it is necessary can be considered to be an "errorless surrogate function swindle", made possible because of the relationship of the kinetic energy to total cost function used in quantum annealing. This can be expected to be a recurring feature in the application of quantum annealing to other combinatorial optimization problems.

4.2.3 Further efficiency gains with look-up tables

The exponential function $\exp(x)$ is much more expensive to compute than division or multiplication [Schraudolph, 1999]. Since the value of $\exp(-\Delta\mathcal{H}/T)$ is required many times, it is desirable to be able to use look-up tables, and avoid the expense of computing the exponential function directly each time [Morgenstern, 1996]. The expression $\exp(-\Delta\mathcal{H}/T)$ is equivalent to $\exp(\Delta KEterm/T)\,.\exp(-\Delta PEterm/T)$. As the temperature T always remains constant in QA-Col, we can pre-compute all possible values of $\exp(-\Delta PEterm/T)$ by noting that if Δ_{max} is the maximum degree of G, then $\Delta\mathcal{H}_p$ can only take on integer values ranging from $-\Delta_{max}$ to $+\Delta_{max}$. This means that an array of size $2\Delta_{max}+1$ is large enough to store all possible values of $\exp(-\Delta PEterm\ /\ T)$.

In order to pre-compute the values of $\exp(\Delta KEterm/T)$, we first determine the range of possible values of $\Delta KEterm$. Substituting $|V_\alpha| = |V|$ and $|V_\beta| = 0$ in (4.7) gives $4(|V| - 1)$ as the maximum value of ΔI_KEterm over all possible pairs of colour class sizes. A similar reasoning gives $4(1 - |V|)$ as the minimum. Therefore an array of size $8|V|$ is needed. This second look-up table needs to be recomputed at the end of every Monte Carlo step because it depends on Γ which decreases. Nevertheless, this is relatively inexpensive, as the number of entries in the lookup table is small in comparison to the number of attempted moves in every Monte Carlo step.

4.3 Implementation Specific Considerations

4.3.1 Pseudo-random number generator

The Metropolis criterion in CA-Col and QA-Col requires a random number generator. In software environments, a function referred to as a pseudo-random number generator (PRNG) is used for this purpose. PRNGs deterministically produce a number on each function call, with the aim of providing a sequence complex enough to appear random to an application. A study in [Maucher et al., 2008] found that the quality of PRNGs have a noticeable effect on the performance of simulated annealing algorithms. One of the main determinants of the quality of a PRNG is its period. (There are several other tests for high quality PRNGs [Marsaglia & Tsang, 2002; Lecuyer & Simard, 2007].) The period of a PRNG determines how many numbers are generated before the sequence repeats. There is a balance to strike between the quality of a PRNG and affordable computational expense. PRNGs can range between simple linear congruential generators to more complex generators such as the Mersenne twister [Matsumoto & Nishimura, 1998].

We experimented with the XOR shift generator presented in [Marsaglia, 2003], the one presented in [Panneton & L'ecuyer, 2005], and the in built-in rand function from MinGW C++, and settled on the generator in [Marsaglia, 2003], as it provided an adequate balance between quality and efficiency.

4.3.2 The dynamic set of conflicting vertices

In both CA-Col and QA-Col, each attempted move requires a selection of a conflicting vertex uniformly at random for a colour change. For an efficient implementation, it is desirable that that a set V_c consisting of all vertices involved in edge conflicts is maintained. Otherwise, one would have to select from V at random and check if it is conflicting, possibly having to discard many non-conflicting vertices in the process. Constant time random access into the data structure of V_c is desirable in order to select a vertex quickly. When a conflicting vertex moves to a different colour class, this usually affects the conflict status of several other vertices, possibly requiring an extensive update to V_c. Therefore, ideally, we want the data structure from which V_c is implemented to be capable of addition and removal in constant time.

The library data structures supporting constant time random access which are available in programming languages, such as vector in C++ and array lists in Java would not usually support constant time addition and removal from an arbitrary position in the data structure. On the other hand, set-like data structures from those same libraries which support fast membership checking, addition and removal do not usually support random access, as they often make use of trees or hash tables. However a custom data structure is able to take into account the specifics of the situation, and provide these seemingly diametrically opposed useful characteristics. Unlike in the general case, all the possible values of V_c are restricted to elements of V, and thus its size is bounded by $|V|$. This enables V_c to be implemented as an associative array with a resizable list backed up by a separate array of fixed-size $|V|$ storing the index of every entry that is actually present in V_c. Any absent vertex in V_c would have a null value (or -1) entered for it in the backing array. This way, addition, removal and membership checking can be done in constant time.

4.3.3 The list of dynamic sets for the colour classes

An important data structure to maintain is a partition of V into colour classes, which are disjoint subsets of V. Fast iteration over a colour class is imperative in the cases where the change in the kinetic energy in (4.6) needs to be calculated in full. Colour classes also need to be updated anytime a colour change of any vertex occurs. This involves addition to and removal from the respective colour classes. One associative array data structure, of the type used for the conflicting vertices in Section 4.3.2 can be set up for each colour class. While this would meet all the time requirements of data accessibility, this basic idea would take up more space than necessary, as each colour class would have a resizable array with a maximum size of $|V|$ and another backing array of size $|V|$ to store the corresponding indices of the present vertices. An improvement involves taking advantage of the mutual exclusivity of the colour classes by using only two backing arrays of size $|V|$ for all of the colour classes: one to keep track of the index of each present vertex in its resizable list, and another to store the colour that each vertex is currently assigned.

4.4 Results and Discussion

4.4.1 Experimental methodology

In order to determine whether quantum annealing improves on classical annealing for the k-colouring problem, CA-Col and QA-Col were implemented as described in Algorithm 4.1 and Algorithm 4.2 respectively in C++ on a PC running Windows XP with a 3.0 GHz processor and 3.0 GB of memory. The stopping criterion was a 5 hour time limit as used in recent graph colouring publications such as [Lü & Hao, 2010] and [Porumbel et al., 2010b]. QA-Col is also compared to the strongest algorithms in the literature by comparing the best upper bounds reached for the chromatic number.

The benchmark graphs used were from the second DIMACS competition [Johnson & Trick, 1996]. We considered the benchmark graphs that have been known to be challenging. Those include Leighton graphs [Leighton, 1979], Erdös-Rényi random graphs [Erdös & Rényi, 1959], geometric random graphs [Penrose, 2003], flat graphs [Culberson et al., 1995; Culberson & Luo, 1996] and a Latin square graph [Lewandowski & Condon, 1996].

Leighton graphs were introduced as part of a study of the application of the graph colouring model to scheduling [Leighton, 1979]. A Leighton graph is generated by planting cliques of various sizes up to a given size by a specified procedure that guarantees that the maximum clique size is the chromatic number, and the density of the graph is never greater than 0.25. A clique is a set of vertices that are all pair-wise connected to one another. It was suggested by [Leighton, 1979] that graphs with a density of about 0.25 frequently occurred in applications of graph colouring to exam scheduling. Real world data supporting this notion has been observed [Lewandowski & Condon, 1996]. The Leighton graphs from the DIMACS benchmarks graphs have the naming format Le$x_\chi y$, where x is the number of vertices, χ is the chromatic number, and y distinguishes between various Leighton graphs.

Erdös-Rényi is a widely used model of generating random graphs [Erdös & Rényi, 1959; Erdös & Rényi, 1960] in which a graph of x vertices is generated by independently including each possible edge with probability p. Erdös-Rényi random graphs from the DIMACS benchmarks follow the naming format of DSJC$x.y$ or C$x.y$, where x is the number of vertices and $y/10$ is equivalent to the probability p.

Geometric random graphs are generated by placing x vertices randomly in a square of unit size, and connecting with an edge any two vertices within a distance d of each other [Penrose, 2003]. The geometric random graphs from the DIMACS benchmarks are of the naming format DSJR$x.y$ or R$x.y$, where x is the number of vertices, and y is equivalent to $d/10$. A "c" is added as a suffix to the name of a graph when the complement of the constructed graph is meant.

Flat graphs are denoted by Flatx_χ_0, where x is the number of vertices, χ is the chromatic number, and "0" is the value of a construction parameter termed the flatness. Details of how these graphs are generated are given in [Culberson & Luo, 1996].

Additionally, there is one Latin square graph Latin_square_10 in the DIMACS benchmarks first provided in [Lewandowski & Condon, 1996] based on the theory of Latin squares. The maximum independent set size in Latin_square_10 is 10, and the graph consists of 900 vertices.

The comparison between CA-Col and QA-Col was done by solving several k-colouring problems with CA-Col and QA-Col, and noting the differences in the number of attempted and accepted moves required for a successful run, as well as the average wall-clock time. Robustness was compared by noting how frequently a solution was successfully obtained. It was also checked whether QA-Col was able to improve on CA-Col by finding colourings with lower values of k than could be obtained with CA-Col. Finally we compared the results of both CA-Col and QA-Col to those of successful algorithms mainly by noting the lowest value of k reached for each graph. This is customary, since other methods of comparison such as the time taken, and the numbers of nodes encountered are known to be problematic, due to varying experimental conditions, algorithms and implementations.

4.4.2 Parameter tuning

CA-Col has two critical parameters that often need to be set differently depending on the problem instance. They are the initial temperature T_0, and the inverse rate of decrease of temperature $MaxSteps$. The multiplier M, which plays a part in determining how many moves are attempted before the temperature is decreased can usually be set large enough that it does not need to change for a variety of graphs. We fixed M to a value of 4 for all problem instances considered in this chapter. In [Johnson et al., 1991], which featured a classical annealing k-colouring algorithm very similar to CA-Col, the initial temperature T_0 was set by choosing a value such that the starting move acceptance ratio was approximately 50-60%. There, the authors noted that most of the improvement in the value of the cost function occurred towards the end of the simulation when the acceptance ratio was much lower. They did not use lower starting acceptance ratios in order not to increase the risk that the algorithm would get trapped in a local minimum, and in order to try and maintain generality amongst a wide variety of instances. However, a classical annealing algorithm with a fixed temperature and a different neighbourhood choice for graph colouring worked with much lower starting acceptance ratios [Morgenstern, 1996].

For CA-Col, we adopted starting temperatures and acceptance ratios that were as low as possible, as an alternative to the approach in [Johnson et al., 1991], and in preparation for the quantum annealing algorithm QA-Col. Quantum annealing is primarily controlled by the field strength Γ, and generally requires low temperatures. A comparison of classical annealing to quantum annealing for the travelling salesman problem in [Martoňák et al., 2004] found that a successful low value for the starting temperature T_0 in classical annealing could be used as an approximation for the resultant temperature PT for quantum annealing. Trial runs with CA-Col and QA-Col confirmed that this usually applied to the graph k-colouring problem as well. In attempting to find a guideline for setting the starting temperature for CA-Col and QA-Col, it was observed that good values of T_0 for a problem instance (G, k) were often suitable for the more difficult problem instance $(G, k-1)$. This means that faster parameter tuning can be done by performing trial runs with high (easy) values of k, and the resulting starting temperature T_0 would still be relevant for lower (more difficult) values of k. Obviously it is important that the high values of k are not so high as to be trivial. For example, the lowest value of k for which the random graph DSJC250.5 has ever been coloured is 28. Especially because of its small size, the instance (DSJC250.5, 30) can be solved very quickly. A successful value of T_0 for (DSJC250.5, 30) can then be used to attempt (DSJC250.5, 29), and then (DJSC250.5, 28). Propagating T_0 values from $k = 30$ through to $k = 28$ is reliable, as there is only a distance of 2 colours. For DSJC250.5, trying to start from a much higher value such as $k = 50$ would be less reliable, especially as the instance (DSJC250.5, 50) is trivial.

Trial runs also play a part in the selection of the value of the starting field strength Γ_0 for QA-Col. Here again, it is important to use higher values of k to speed up the tuning. As the problem instances from a given graph G get more difficult by the reduction of k, it was observed that the starting value Γ_0 could usually just be incremented by a few hundredths. Tests demonstrated that the ranges $PT_0 \in (0,1]$ and $\Gamma_0 \in (0, 3PT_0]$ are appropriate for many graphs in the DIMACS benchmark.

4.4.3 Experimental results

Table 4.1 shows the results of the experiments with CA-Col. Column 1 (Graph) contains the names of the graphs under consideration from the DIMACS benchmarks and the lowest known value k* in the literature; column 2 (k) is for the number of available colours; column 3 (T_0) contains the value for the initial temperature; column 4 ($MaxSteps$) contains the maximum number of steps; column 5 (Attempted) and column 6 (Accepted) contain the number of attempted and accepted moves respectively, both averaged over successful runs; column 7 (t) is for the average wall-clock time for the successful runs out of ten, and the hit rate or success ratio out of ten runs is contained in column 8 (Hit).

Table 4.1: CA-Col results with a 5 hour time limit

Graph(k^*)[a]	k	T_0	$MaxSteps$	Attempted	Accepted	t[s]	Hit
DSJC250.5(28)	28	0.35	1.0×10^5	5.0×10^8	1.6×10^7	45	7/10
DSJC500.1(12)	12	0.45	1.0×10^7	4.8×10^9	4.1×10^8	489	10/10
DSJC500.5(48)	49	0.35	1.0×10^6	2.1×10^{10}	3.8×10^8	2117	8/10
	48	0.35	1.0×10^6	-	-	-	0/10
DSJC500.9(126)	127	0.20	1.0×10^6	2.4×10^{10}	2.6×10^8	2330	10/10
	126	0.20	1.0×10^7	-	-	-	0/10
Le450_15c(15)	15	0.60	1.0×10^6	6.3×10^8	6.3×10^7	73	10/10
Le450_15d(15)	15	0.60	1.0×10^6	3.7×10^8	4.2×10^7	41	4/10
Flat300_28_0(28)	31	0.35	1.0×10^6	3.9×10^9	1.1×10^8	375	10/10

[a]See the footnote of Table 4.2 with regards to k^* values

Each of the ten runs for each problem instance (G, k) was done with a different seed to the random number generator.

The main results of QA-Col are presented in Table 4.2. Column 1 (Graph) contains the names of the graphs and best known results; column 2 (k) contains the number of available colours; column 3 (PT) contains the values for the fixed resultant temperature PT; column 4 (Γ) contains the initial values of the field strength Γ, column 5 ($MaxSteps$) contains the maximum number of Monte Carlo steps to run the algorithm for. Columns 6 (Attempted) and 7 (Accepted) contain the number of moves that were attempted and accepted respectively, averaged over successful runs; column 8 (ES%) contains the percentage of attempted moves for which we avoided having to fully calculate the actual kinetic energy change after the errorless swindle (ES) technique introduced in Section 4.2.2; column 9 (t) contains the average wall-clock time of the number of successful runs; and column 10 (Hit) contains the success rate of each instance out of 10 runs.

Table 4.2: QA-Col results with a 5 hour time limit

Graph(k^*)[a]	k	PT	Γ_0	$MaxSteps$	Attempted	Accepted	ES[%]	t[s]	Hit
DSJC250.5(28)	28	0.35	0.75	1.0×10^4	6.1×10^7	2.1×10^6	95	8	10/10
DSJC500.1(12)	12	0.45	1.30	1.0×10^6	4.5×10^8	3.8×10^7	88	82	10/10
DSJC500.5(48)	49	0.35	0.65	1.0×10^5	4.3×10^8	9.3×10^6	96	63	10/10
	48	0.35	0.75	1.0×10^5	3.4×10^9	7.0×10^7	97	494	10/10
DSJC500.9(126)	127	0.20	0.35	1.0×10^5	7.9×10^8	8.8×10^6	99	103	10/10
	126	0.20	0.38	1.0×10^6	9.9×10^9	1.1×10^8	99	1198	10/10
DSJC1000.1(20)	20	0.44	1.10	1.0×10^6	9.1×10^9	3.8×10^8	85	1951	9/10
DSJC1000.5(83)	84	0.36	0.68	2.0×10^7	1.8×10^{10}	2.8×10^8	97	2842	10/10
	83	0.36	0.72	5.0×10^7	8.2×10^{10}	1.0×10^9	98	12,773	9/10
DSJC1000.9(223)	223	0.23	0.38	1.0×10^8	2.6×10^{10}	2.2×10^8	99	4100	8/10
	222	0.19	0.375	2.0×10^9	1.1×10^{11}	5.6×10^8	99	13,740	2/10
Le450_15c(15)	15	0.60	1.60	1.0×10^5	1.9×10^7	1.9×10^6	83	4	10/10
Le450_15d(15)	15	0.60	1.80	1.0×10^5	1.5×10^8	1.3×10^7	86	26	10/10
Le450_25c(25)	26	0.30	0.48	1.0×10^5	5.5×10^7	2.2×10^6	74	9	10/10
	25	0.30	0.58	2.0×10^9	3.7×10^{10}	1.6×10^9	86	5592	2/10
Le450_25d(25)	26	0.30	0.48	1.0×10^5	8.7×10^7	2.8×10^6	74	13	10/10
	25	0.30	0.59	1.0×10^7	6.7×10^{10}	3.2×10^9	87	10,069	1/10
Flat300_28_0(28)	31	0.35	0.75	1.0×10^5	1.5×10^8	4.7×10^6	95	19	10/10
Flat1000_76_0(82)	83	0.36	0.67	5.0×10^7	1.4×10^{10}	1.9×10^8	97	2250	10/10
	82	0.36	0.71	5.0×10^7	6.4×10^{10}	7.9×10^8	97	9802	7/10
R1000.5(234)	239	0.11	0.07	1.0×10^5	2.3×10^{10}	5.7×10^8	85	5979	10/10
	238	0.11	0.07	1.0×10^8	3.7×10^{10}	8.9×10^8	86	9511	3/10
DSJR500.5(122)	123	0.14	0.10	1.0×10^8	2.4×10^9	8.1×10^7	78	483	10/10
	122	0.15	0.10	1.0×10^8	1.8×10^9	5.6×10^7	73	370	10/10
DSJR500.1c(85)	85	0.25	0.55	1.0×10^6	3.4×10^9	6.1×10^7	97	525	2/10
R250.5(65)	65	0.10	0.10	1.0×10^6	1.1×10^9	5.8×10^7	92	168	9/10
R1000.1c(98)	98	0.50	1.50	1.0×10^6	1.6×10^9	1.8×10^7	98	287	10/10
Latin_square_10(98)	98	0.45	0.90	1.0×10^7	9.0×10^9	9.9×10^7	98	1449	10/10

[a]The values for k^* in the table were the best in the literature before our paper [Titiloye & Crispin, 2011a] was accepted for publication. Some of these were subsequently improved as a result of our further work presented in chapter 5 and 6 and published in [Titiloye & Crispin, 2011b] and [Titiloye & Crispin, 2012] respectively. More information about the origin of the k^* values can be found in [Lü & Hao, 2010].

The maximum number of steps, $MaxSteps$ for both QA-Col and CA-Col was selected to be large in proportion to the size of the graph, and the known difficulties of other algorithms in solving them, in order to create a slow annealing schedule with increased chances of success. To facilitate comparisons between QA-Col and CA-Col, the value for $MaxSteps$ in CA-Col was set to be 10 times that of QA-Col, to provide equivalence by accounting for the fact that QA-Col used a population of $P = 10$. Just as in the experiments with the travelling salesman problem in [Martoňák et al., 2004], good values for the initial temperature T_0 for CA-Col tended to be good values for PT for QA-Col.

After experimenting with CA-Col for a number of k-colouring instances, it was apparent that it was no better than its predecessor in [Johnson et al., 1991], which mainly differs from CA-Col in its parameter tuning approach. Particularly, unlike the current powerful metaheuristics for graph colouring, CA-Col could not find any 48-colourings for DSJC500.5. QA-Col was able to do this easily. Even on the smaller graphs such as DSJC250.5 where CA-Col matched QA-Col in attaining the current best results of 28-colourings, CA-Col required much more computational effort in terms of the number of attempted moves, as is apparent from the entry of DSJC250.5 in Table 4.1 and Table 4.2. This indicates that CA-Col is often less effective than QA-Col in escaping locally minimal traps. Experimenting with different values of the parameters could not improve the worse performance of CA-Col with respect to QA-Col. It can be seen from Table 4.2 that QA-Col found a new upper bound of 222 for the chromatic number of DSJC1000.9. On the other 18 graphs, the best known results in the literature were matched except in the two cases of Flat300_28_0 and R1000.5. The algorithms that do best on these two graphs are known to be specialized to them. For example, the partial colouration algorithm in [Blöchliger & Zufferey, 2008] for the flat graph Flat300_28_0 and the column generation algorithm in [Malaguti et al., 2008] for R1000.5.

4.4.4 The influence of the errorless swindle on computational speed

It was shown analytically in Section 4.2.2 that the computational expense of calculating the change in the Hamiltonian $\Delta\mathcal{H}$ can be reduced throughout the running of QA-Col by first calculating a lower bound $LB\Delta\mathcal{H}$, and only calculating the actual value of $\Delta\mathcal{H}$ in relatively few cases. This was termed an errorless swindle in Section 4.2.2. An important feature of this reduction in computational expense is that the colouring configurations that QA-Col visits are not changed in any way, and no errors or deviations are introduced as a result of its use.

We now experimentally demonstrate the speed-up it provides, its relationship to the density of the graph, and that it does not introduce any errors. The graph k-colouring instances used are (DSJC1000.1, 20), (DSJC1000.5, 83) and (DSJC1000.9, 223). Provided that the seed of the random number generator is kept the same, QA-Col should perform the same operations on multiple runs for the same problem instance, if all other things are kept constant. Therefore, each of the problem instances is run twice: once with the errorless swindle turned off, and a second time with the errorless swindle turned on. There was a reduction in wall-clock time of 61% for (DSJC1000.1, 20), 58% for (DSJC1000.5, 83), and 47% for (DSJC1000.9, 223) when errorless swindle was turned on. Additionally, as long as the seed for the random number generator was kept the same, QA-Col visited the same configurations and ended up with exactly the same results irrespective of whether the errorless swindle was turned on or off. The results also suggest that the sparser graphs benefit the most.

4.4.5 The influence of the choice of the PRNG

The XOR shift generator with only three shifts presented in [Marsaglia, 2003] was used in all of our experiments. The same type of generator with seven shifts in [Panneton & L'ecuyer, 2005] did not provide better results despite its larger period, and only resulted in longer computing times. A geometric random graph R250.5 could not be coloured with 65-colours by QA-Col when the built-in rand function available from the MinGW C++ compiler was used, while the XOR shift generators did not have this problem. This drawback of the built-in generator was not so visible on other graphs.

The adverse effect that weaker random number generators have on R250.5 can be at least partially explained by the observation that a larger proportion of zero-cost moves (where $\Delta\mathcal{H}_p = 0$) were encountered in comparison to other graphs. It is therefore reasonable that a strong random number generator is desirable in order to avoid getting stuck looping on a plateau consisting of a large number of configurations with the same potential energy.

4.4.6 Comparison with alternative algorithms

As CA-Col has been demonstrated to be a worse algorithm than QA-Col in Section 4.4.3, it is important to focus on QA-Col, and how it measures up to the leading graph colouring algorithms. Table 4.3 facilitates this comparison. The results from [Johnson et al., 1991] are in column 4 as "Joh1991"; those of [Morgenstern, 1996] are in column 5 as "Mor1996"; those of [Galinier & Hao, 1999] are in column 6 as "GH1999"; those of [Funabiki & Higashino, 2000] are in column 7 as "FH2000"; those of [Galinier et al., 2008] are in column 8 as "GHZ 2008"; those of [Malaguti et al., 2008] are in column 9 as "MMT2008"; those of [Blöchliger & Zufferey, 2008] are in column 10 as "BZ 2008"; those of [Lü & Hao, 2010] are in column 11 as “LH2010”; and those of [Porumbel et al., 2010b] are in column 12 as “PHK2010”.

The Joh1991 results from [Johnson et al., 1991] are a selection of the best solutions obtained from a wide-ranging study mainly consisting of various methods of applying classical annealing to graph colouring, as well as some greedy and exhaustive approaches. The classical annealing variants considered included a penalty function approach, a kempe-chain annealing approach and most importantly because of the relevance to this work, a k-colouring approach.

Table 4.3: A comparison between QA-Col and some competitive algorithms

Graph(k^*)[a]	QA	CA	Joh	MOR	GH	FH	GHZ	MMT	BZ	LH	PHK
	-Col	-Col	1991	1996	1999	2000	2008	2008	2008	2010	2010
DSJC250.5(28)	28	28	29	28	28	28	28	28	-	28	28
DSJC500.1(12)	12	12	13	12	-	12	12	12	12	12	12
DSJC500.5(48)	48	49	49	48	48	49	48	48	48	48	48
DSJC500.9(126)	126	127	128	126	-	127	126	127	126	126	126
DSJC1000.1(20)	20	-	21	21	20	21	20	20	20	20	20
DSJC1000.5(83)	83	-	86	84	83	88	84	83	89	83	83
DSJC1000.9(223)	**222**	-	226	226	224	228	224	224	225	223	223
Le450_15c(15)	15	15	-	15	15	15	15	15	15	15	15
Le450_15d(15)	15	15	-	15	-	15	15	15	15	15	15
Le450_25c(25)	25	-	-	25	26	26	26	25	25	25	25
Le450_25d(25)	25	-	-	25	-	26	26	25	25	25	25
Flat300_28_0(28)	31	31	-	31	31	31	31	31	28	29	29
Flat1000_76_0(82)	82	-	-	89	83	87	84	82	87	82	82
R1000.5(234)	238	-	-	241	-	237	-	234	247	245	237
DSJR500.5(122)	122	-	124	123	-	122	125	122	125	122	122
DSJR500.1c(85)	85	-	85	85	-	85	86	85	85	85	85
R250.5(65)	65	-	-	65	-	65	-	65	65	65	65
R1000.1c(98)	98	-	-	-	-	98	-	98	98	98	98
Latin_square_10(98)	98	-	-	98	-	99	105	101	-	99	98

[a]See the footnote of Table 4.2 with regards to k^* values

Apart from parameter tuning, the k-colouring classical annealing algorithm in [Johnson et al., 1991] is very similar to our CA-Col, as well as the algorithm in an earlier classical annealing study for k-colouring in [Chams et al., 1987]. The random graphs and geometric random graphs which became part of the DIMACS benchmarks were first used in [Johnson et al., 1991]. The paper was also one of the first to present graph colouring results from a stochastic local search algorithm, producing results that were the best in its time. From Table 4.3, it can be seen that on the first four random graphs, CA-Col slightly outperforms Joh1991, probably due to a different tuning approach of starting with low acceptance ratio, whereas QA-Col clearly outperforms Joh1991 by saving one colour on each of the four instances.

The Mor1996 results from [Morgenstern, 1996] are from a classical annealing algorithm with some design decisions that differ from [Johnson et al., 1991] on some important points. Low starting acceptance ratios were aimed for. This meant setting low starting temperatures which allowed a strategy in which decrementing the starting temperature was unnecessary. Therefore in contrast to [Johnson et al., 1991], experiments were carried out with a low fixed temperature. Although applications of classical annealing with a low fixed temperature are not very common in the literature, they are studied in [Connolly, 1990; Cohn & Fielding, 1999; Fielding, 2000]. The algorithm in [Morgenstern, 1991] used a technique similar to that presented in [Greene & Supowit, 1986] to reduce efficiency issues arising from the rejection of a large proportion of bad moves. The main neighbourhood used was termed impasse-class. The impasse-class strategy maintains k colour classes that together always form a proper colouring using a subset of V, as well as one impasse class consisting of vertices that cannot yet enter into any of the k colour classes without creating edge conflicts. The aim is then to empty the impasse class with exchange moves. Some of the good results on the random graphs were obtained by initialising the classical annealing algorithm not to a random state, but with colourings obtained by a semi-exhaustive independent set extraction procedure called XRLF presented in [Johnson et al., 1991]. Highlights of [Morgenstern et al., 1996] include being the first to present 48-colourings for DSJC500.5 and 84-colourings for DSJC1000.5. Up until the appearance of the Evolutionary-Tabu algorithm in [Porumbel et al., 2010b] and our quantum annealing results in [Titiloye & Crispin, 2011a] it was also the only reporting of a 98-colouring for the Latin square graph Latin_square_10. QA-Col outperforms Mor1996 on the larger random graphs and geometric random graphs and matches it on all other graphs.

The GH1999 results are from an Evolutionary-Tabu algorithm in [Galinier & Hao, 1999]. Unlike an earlier Evolutionary-Tabu algorithm in [Fleurent & Ferland, 1996], the approach in [Galinier & Hao, 1999] and its precursor [Dorne & Hao, 1998b] carry out recombination by blending colour classes in a way that respects the notion that graph colouring is a grouping problem, and not only an assignment problem. This is rather like the care that was taken in Section 4.1.1 by selecting a Boolean variable system for graph colouring that does not suffer from obvious symmetry problems while the individuals in a population are cooperating. At the time, three new results of a 20-colouring for DSJC1000.1, 83-colouring for DSJC1000.5 and 224-colouring for DSJC1000.9 were reported in [Dorne & Hao, 1998b]. These were also replicated in [Galinier & Hao, 1999] in addition to 83-colourings for Flat1000_76_0 that were new at that time. In terms of the quality of results, it can be seen from Table 4.3 that QA-Col is able to compete with GH1999.

The FH2000 results are obtained from a local search algorithm in [Funabiki & Higashino, 2000]. It finds some of the best results available on the geometric random graphs.

The GHZ2008 results are from an adaptive memory approach in [Galinier et al., 2008]. The idea of an adaptive memory algorithm for graph colouring was inspired by an earlier algorithm for the vehicle routing problem in [Rochat & Taillard, 1995], which stored components of solutions that could be combined to form a full solution to be improved by local search. As far as graph colouring is concerned, the adaptive memory approach is an interesting alternative design for a hybrid evolutionary algorithm that does not appear to be as strong as the more traditional design in [Galinier & Hao, 1999], considering the relatively weaker results on graphs such as DSJC1000.5. Quantum annealing is able to match and improve upon all results presented in [Galinier et al., 2008].

The MMT2008 results are from an Evolutionary-Tabu algorithm in [Malaguti et al., 2008]. The algorithm is combined with an exact lower-bounding procedure that is sometimes able to prove optimality in reasonable time. The algorithm reported 82-colourings for the first time for the graph Flat1000_76_0. It was also the first to prove that the chromatic number of R1000.5 is 234. Quantum annealing is able to match all the MMT2008 results except for 234-colourings on R1000.5. Quantum annealing also improves on MMT2008 on some graphs such as DSJC1000.9 and Latin_square_10.

The BZ2008 results are from a Tabu search algorithm [Blöchliger & Zufferey, 2008]. It uses a variation of the impasse-class neighbourhood [Morgenstern, 1996] and was the first to produce 28-colourings for the graph Flat300_28_0. As can be seen from Table 4.3, quantum annealing competes well with this algorithm.

The LH2010 results are from an Evolutionary-Tabu algorithm in [Lü & Hao, 2010], which improves upon the algorithm in [Galinier & Hao, 1999] mainly by using multi-parent recombination operators and controlling the distance between individual members of a population. It is strong on the Leighton graphs, colouring all of them optimally with a 100% success rate and in very fast times compared to competing algorithms. It is able to replicate the 82-colourings of [Malaguti et al., 2008] for Flat_1000_76_0 like quantum annealing. The algorithm is able to find 223-colourings for DSJC1000.9 as were independently found in [Xie & Liu, 2009], and also improves on known results for two larger random graphs C2000.5 and C4000.5 that are addressed in chapter 5 and chapter 6 of this thesis. As can be seen from Table 4.3, quantum annealing competes well with this algorithm.

The PHK2010 results are from an Evolutionary-Algorithm in [Porumbel et al., 2010b], which also extends ideas in [Galinier & Hao, 1999], mainly with multi-parent recombination operators and a control of the distance between the members of a population. However there are some structural differences in the design of its Evolutionary component as well as the features of its underlying Tabu search. It uses a population spacing technique [Porumbel et al., 2011], which proves applicable to an enhanced version of quantum annealing in chapter 5. In [Porumbel et al., 2010b], 98-colourings were reported only for the second time in the literature since [Morgenstern et al., 1996]. Our quantum annealing became the third to do this [Titiloye & Crispin, 2011a]. The PHK2010 results also include 223-colourings for DSJC1000.9, which quantum annealing is able to improve upon by finding 222-colourings. PHK2010 also features results for the larger random graphs C2000.5 and C4000.5, matching the results in [Lü & Hao, 2010] and improving on C4000.5 by saving one colour. As can be seen from Table 4.3, quantum annealing competes well with this algorithm.

A contemporaneous attempt at a quantum annealing implementation for graph colouring by [Lecina, 2011] reported the negative conclusion that quantum annealing was outperformed by classical annealing. This publication appeared shortly after our journal paper [Titiloye & Crispin, 2011a] on which parts of this chapter are based. The definition of the spin variables used by [Lecina, 2011] was derived from a k-state Potts spin model for graph colouring which is well known in the physics community. However a direct usage of the Potts spin model in a population-based algorithm results in symmetry problems, as the names of the colours take on an unwarranted importance. An interdisciplinary approach that considered the success and failures of past Evolutionary-Tabu algorithms has allowed us to avoid this pitfall by introducing a more involved spin variable definition that avoids graph independent symmetry problems.

4.5 Conclusion

The aim of this chapter was to design a competitive Monte Carlo quantum annealing algorithm for graph colouring and to study the characteristics and specializations such an algorithm needs to possess. Although there exist past quantum annealing studies for various problems including the travelling salesman problem [Martonák et al., 2004] and Boolean satisfiability [Battaglia et al., 2005] showing some promise in competing with classical annealing, they have generally been unable to compete with state of the art algorithms for the problem at hand. The present study set out to change that state of affairs. Each application of quantum annealing to a different combinatorial optimization problem requires some specialization especially in representing the problem as a spin model and in the choice of the neighbourhood moves to be performed during the search. An important feature of the graph colouring quantum annealing algorithm presented is the choice of the definition of a set of constrained spin variables that treat graph colouring as a grouping problem [Falkenauer, 1994], and prevent the algorithm from being severely affected by symmetry problems that arise due to the redundancy of colour naming. A different choice of the definition of spin variables was at least partly responsible for the negative conclusions reported by [Lecina, 2011] in a contemporaneous study of the application of quantum annealing to graph colouring.

In order to achieve a competitive running time in comparison to the leading algorithms, the change in kinetic energy needed to be computed as efficiently as possible. This was made more difficult by the intricate definition of our spin variables. An incremental technique to calculate the kinetic energy change was devised. Additionally an errorless “surrogate cost function swindle” [Tovey, 1988] was provided based on an upper bounding of the kinetic energy change. This meant that the cost of computations could be reduced even further.

Experiments have shown that quantum annealing outperforms classical annealing on the graph colouring problem. Quantum fluctuations defined by a population based kinetic energy, and controlled by a field strength parameter were effective in allowing the quantum annealing algorithm to tunnel to favourable states that are difficult for classical annealing to attain by thermal fluctuations alone. Moreover, quantum annealing proves competitive with the leading graph colouring algorithms which are mainly Evolutionary-Tabu hybrid algorithms. Most of the best results on challenging graphs from the DIMACS benchmarks are matched, and quantum annealing even finds a new result of 222-colourings for DSJC1000.9. The results of this chapter indicate that quantum annealing can be a viable approach for population-based stochastic local search algorithms that can be used in place of traditional Evolutionary algorithms. The study enhances our understanding of how to adapt quantum annealing to grouping problems such as graph colouring, and provides the hope of extensibility to similar problems.

A limitation of most implementations of quantum annealing, including the current one, is the relative difficulty in determining suitable tuning parameters in comparison to Evolutionary-Tabu algorithms. Studies in later chapters of this thesis will focus on parameter tuning issues and seek improvements. Additionally, the monitoring of the partition distance [Gusfield, 2002] between individuals has proved useful for preventing premature convergence in the leading Evolutionary-Tabu algorithms for graph colouring. An adaptation of this idea to quantum annealing is investigated in the next chapter.

Chapter 5

Enhancements and Variations to the Quantum Annealing Algorithm

Following the introduction of a quantum annealing algorithm for graph colouring in the last chapter, the present one deals with investigations of enhancements and variations aimed at making the algorithm even more competitive.

Section 5.1 is concerned with a simplification of parameter tuning by not decreasing the field strength during the course of the simulation as is customary [Santoro et al., 2002; Das & Chakrabarti, 2008], but heuristically choosing its value so that it can be kept fixed throughout. In Section 5.2, the partition distance [Gusfield, 2002] is used to space the replicas in order to combat premature convergence. This is inspired by its success in the different domain of Evolutionary-Tabu algorithms for graph colouring [Porumbel et al., 2010b]. It marks the first time that such a replica spacing mechanism has been incorporated into any quantum annealing algorithm. Section 5.3 addresses the parallelization of our graph colouring algorithm in order to take advantage of more processing power in tackling this computationally intensive problem. Only sequential versions of the leading graph colouring metaheuristics [Malaguti et al., 2008; Lü & Hao, 2010; Porumbel et al., 2010b] are available even though computational times can run into several weeks for some difficult instances. We present a parallelized graph colouring algorithm, which also incorporates the enhancements in Section 5.1 and Section 5.2.

Results and discussions are contained in Section 5.4. In Section 5.4.1 and Section 5.4.2, empirical results for the enhanced quantum annealing algorithm are presented. Section

5.4.3 is concerned with comparisons with other algorithms in the literature. These are favourable for quantum annealing. Section 5.4.4 discusses the influences of the individual enhancements. Section 5.4.5 discusses variants of our algorithm, which deal with some peculiar instances effectively.

5.1 The Fixed-Γ Quantum Annealing

The initial quantum annealing algorithm for graph k-colouring presented in chapter 4 required three critical and problem instance specific parameters to be set: a fixed temperature T, field strength Γ, and the rate of decrease of Γ. The standard approach to annealing algorithms such as classical annealing (CA) and quantum annealing (QA) involves the provision of a main control parameter that is decreased over time. In the case of CA, this parameter is the temperature T. In chapter 4, we used a linear annealing schedule to progressively decrease Γ throughout the course of a simulation, just as in [Santoro et al., 2002; Martonák et al., 2004]. The alternative approach of keeping the main parameter heuristically set and fixed throughout the duration of an annealing algorithm has previously been investigated for CA [Connolly, 1990; Cohn & Fielding, 1999; Fielding, 2000]. Fixed-temperature CA, also known as the Metropolis algorithm, has delivered competitive results for the quadratic assignment problem, travelling salesman problem and graph partitioning [Fielding, 2000]. Moreover, it was indicated by [Cohn & Fielding, 1999] that under certain conditions, a fixed-temperature classical annealing algorithm converges to a global minimum in finite time following from the classical theory of Markov chains.

It is interesting to consider whether Γ can be heuristically set for a given graph k-colouring problem instance in such a way that no decrease is needed throughout the duration of the simulation. The main motivations for introducing a fixed-Γ Monte Carlo quantum annealing algorithm are to simplify its tuning and make it more robust. In chapter 4, the rate of decrease of Γ was set by specifying $MaxSteps$, the number of steps it would take to linearly decrease the initial value of Γ to zero. The value of $MaxSteps$ can be considered as an inverse rate of decrease of Γ. In order to derive a practical algorithm, the initial value of Γ was not simply set to an arbitrarily high value, but to a low heuristic value that fit the particular problem instance. If the value of $Maxsteps$ was too small, causing the rate of decrease of Γ to be too fast, quantum annealing usually got trapped by poor local minima. It is particularly important in the graph k-colouring problem to reach a global minimum, as it is only then that the graph can be properly labelled with k colours.

Some of the more challenging problem instances addressed in chapter 4 required such a large value for $MaxSteps$ that Γ changed only very little during the execution of the quantum annealing algorithm. This suggests that we can consider setting a value of Γ for a particular instance, in such a way that no decrease is necessary at all. The algorithm can then be terminated after passing an alternative criterion such as a time limit, or when a global minimum is located. This would mean that only two critical parameters, T and Γ would need to be specified for each problem instance. The experimental results from this change are reported in Section 5.4, and they show that it is helpful in making quantum annealing more robust while simplifying its tuning.

5.2 Quantum Annealing with Replica Spacing

A Monte Carlo quantum annealing algorithm maintains P different sets of configurations or replicas, as depicted in the Hamiltonian (4.5) and Algorithm 4.2 from chapter 4. These replicas are connected in a ring and interact with one another according to a shared component or the kinetic energy, which is the Γ-dependent term in (4.5). As the replicas interact, they become increasingly similar to each other. While this mechanism allows the replicas to jointly navigate the search space and escape shallow local minima, it is obvious that if all replicas were to prematurely become too similar, the effectiveness of the algorithm would be compromised. In the context of the k-colouring problem, a useful measure of how different two colouring configurations are to each other is known as the partition distance [Gusfield, 2002]. If one replica consists of a partition of V given by $C = C_1, C_2, \dots, C_k$ and another replica consists of a partition given by $C' = C'_1, C'_2, \dots, C'_k$, then the partition distance $D(C, C')$ is the minimum number of vertices that must change their class in C, so that C becomes equivalent to C'. In order to compute the partition distance $D(C, C')$, we first define a matrix $M(C, C')$ of order k by k, and assign each entry M_{ij} to the number $|C_i \cap C'_j|$. As explained by [Gusfield, 2002], solving the assignment problem on the matrix $M(C, C')$ in polynomial time gives a number $A(C, C')$, and the required partition distance $D(C, C')$ is given as $|V| - A(C, C')$, where V is the vertex set. The assignment problem can be solved in $O(k^3)$ with the Hungarian algorithm [Kuhn 1965; Jonker & Volgenant, 1986; Wright, 1990].

The distance measure $D(C, C')$, which ranges between 0 and $|V|$ is useful in detecting when any two colourings are too similar to each other, and has been used in some successful Evolutionary-Tabu algorithms for graph colouring [Lü & Hao, 2010; Porumbel et al., 2010b]. A possible use for the partition distance in quantum annealing is to prevent any replica ρ from getting too close to its two directly connected replicas b and f with which it shares spin products. To quantify what it means for the colouring configuration of one replica to be "too close" to that of another replica, we look to recent empirical studies of the spatial arrangement of high quality colouring configurations in the search space. In [Porumbel et al., 2010a], experimenting with a wide variety of graphs, several k-colouring configurations from each graph were generated with Tabu search, and represented within a scaled three-dimensional space. The findings were that high quality colouring configurations (with low numbers of edge conflicts) were arranged in several clusters, with each cluster having a size of approximately $R = |V|/10$. A successful Evolutionary-Tabu algorithm called Evo-Div was designed by the same authors, in which individuals of the population were restricted to a minimum spacing of R between themselves [Porumbel et al., 2010b]. The authors verified that a smaller minimum spacing distance of $R/2$ was too restrictive, while a larger value of $2R$ created too much diversity, and affected convergence adversely.

Previous quantum annealing algorithms have not attempted to control the spacing of replicas, even though the problem of maintaining diversity amongst replicas had been recognised and acknowledged. For instance, in a study of the application of quantum annealing to the Boolean satisfiability [Battaglia et al., 2005], the authors noted that over the course of the execution of the algorithm, the replicas tended to collapse into a set of identical states prematurely. Their attempted remedy was to start increasing Γ to a predetermined maximum any time the system was trapped, and then to start decreasing it afterwards in what was termed "field cycling". This achieved only limited success, but there were other possible responsible factors such as the structure of the potential energy landscape of the combinatorial optimization problem that was being investigated.

Experiments in Section 5.4 with an enhanced version of quantum annealing incorporating replica spacing showed that this idea improved the algorithm considerably. Some test runs indicated that maintaining a minimum spacing distance of $R = |V|/10$ empirically derived by [Porumbel et al., 2010a] for Tabu search was suitable for quantum annealing as well.

5.3 Parallelizing Quantum Annealing

Graph colouring is computationally intensive, with some problem instances from the DIMACS benchmarks requiring several weeks of wall-clock time on a single processor [Lü & Hao, 2010; Porumbel et al., 2010b]. Yet the current leading graph colouring publications [Malaguti et al., 2008; Lü & Hao, 2010; Porumbel et al., 2010b] have not addressed parallelization. Even for combinatorial optimization problems other than graph colouring, it has been pointed out that research on parallel metaheuristics significantly lags that of serial metaheuristics [Crainic & Toulouse, 2003; James et al., 2009]. It is often not the case that inherently sequential algorithms can be trivially converted to parallel algorithms without significant modifications, even if some of these are population based. Given that most end-users now have access to personal computers with multi-core processors, it seems justified that parallelized versions of algorithms for problems as computationally intensive as graph colouring should be addressed more thoroughly by researchers.

5.3.1 Parallelization in the context of metaheuristics

Research on parallel metaheuristics has been grouped into three categories by [Crainic & Toulouse, 2003]. Type 1 is mainly concerned with converting non population based sequential metaheuristics, such as classical annealing, into parallel ones by methods such as parallelizing the evaluation of a computationally intensive cost function, or the dividing of the exploration of the neighbourhood amongst available processors. With Type 2, the problem instance itself is broken up into regions which are divided amongst the processors. Type 3 applies to population based approaches where individuals divided amongst processors run their own local search exploring their own neighbourhoods, while communicating with each other periodically or not at all.

Approaches using Type 1, also known as low level parallelism, aim to follow the sequential execution of an underlying non population based heuristic as closely as possible, while exploiting opportunities for data parallelism and task parallelism. In data parallelism [Hillis & Steele, 1986], the same operations are performed on multiple data. In task parallelism [Subhlok et al., 1993], various tasks are distributed amongst processors. Classical annealing has incorporated Type 1 parallelism by two methods known as single-trial and multiple-trial parallelism [Eglese, 1990]. The single-trial parallelism version of simulated annealing proceeds with all the processors combining to evaluate the cost function and make one move, therefore behaving exactly like the sequential version. Since the opportunity for parallelism in this case is usually limited to the evaluation of the cost function, it can be difficult to attain significant speedups. This is one of the reasons why multiple-trial parallelism has been tried. In multiple-trial parallelism, each processor independently tries to find an acceptable move for a shared configuration. As soon as one of the processors finds a move, the shared configuration is locked while it is being updated, to stop concurrent updates in order to prevent errors. After the update, the new configuration is broadcast to all the processors which repeat the process by trying to find the next move. When the move acceptance ratio is low, as it would be in the case of low temperatures, then a significant speedup may be attained due to less synchronization. However in the case of high move acceptance ratios, the performance can deteriorate due to the excessive periods of time spent by the processors waiting for each other to update the configuration.

Type 2 parallelism involves the splitting of the variables of the configuration of the problem between the processors. For example, the cities of the travelling salesman problem may be divided amongst the processors which perform simulated annealing on their assigned cities, with the aim of combining the results afterwards [Felten et al., 1985].

Type 3 parallelism refers to the case whereby the algorithm is already population based, and each processor possesses its own configuration (or candidate solution). There may or may not be interaction between the individuals in the population. In other words, the individuals may be independent or cooperative. In the context of classical annealing algorithms, this would mean each individual running a separate instance of classical annealing, probably with a different initial configuration, with or without the same parameter values. The Type 3 strategy was called p-control in [Crainic et al., 1997], and multiple-walk in [Verhoeven & Aarts, 1995]. Given that quantum annealing consists of interacting replicas, which can be viewed as individuals in a population, Type 3 parallelism is the most applicable to it.

5.3.2 Parallelization in the context of graph colouring

Even though existing parallel graph colouring algorithms from the literature are not the best performing ones in terms of the quality of results, a review of them is still instructive. An approach hybridizing an exact algorithm with a heuristic algorithm was presented by [Lewandowski & Condon, 1996]. A master process initiates independent processes, each with their own different initial colouring. Some of the processes use a branch and bound colouring algorithm, while the others use an iterative improvement heuristic. The cooperation between the processes is very limited. If any of the processes finds a colouring that results in a new upper bound for the chromatic number, it sends this to the master process that then broadcasts the message to all processes. A fraction of the processes might also occasionally abandon their own configurations and start from a perturbed version of the best found so far.

Due to the high independence of the processors, and the fact that each one works on its own colouring, this implementation qualifies as a Type 3 under the classification of [Crainic & Toulouse, 2003]. However, while it is often advantageous to minimize the time spent by the processors in communication, the type of information shared and how this is used can be very important. In this particular hybrid parallel graph colouring algorithm, the cooperation is basic. While the algorithm benefits from an increase in speed, it does not improve significantly over sequential algorithms in terms of the quality of the final colourings. The parallelization approach in [Morgenstern, 1996] termed Distributed Coloration Neighbourhood Search (DCNS) is similar to the one just described. Additional parallel graph colouring algorithms include the one by [Jones & Plassmann, 1993] based on cooperating to find independent sets; that of [Gebremedhin & Manne, 2000] based on greedy colouring heuristics; [Kokosinski et al., 2004] which employs recombination operators; and [Boman et al., 2005] based on an iterative improvement idea.

5.3.3 A parallel graph colouring quantum annealing algorithm

By representing the P replicas as agents that cooperate via a shared kinetic energy in order to minimize the Hamiltonian $\mathcal{H}$ from (4.5), we present in Algorithm 5.1 a distributed quantum annealing algorithm for k-colouring, which also incorporates the fixed-Γ and replica spacing from Section 5.1 and Section 5.2 respectively. The algorithm can either be considered to be parallelized or distributed depending on the hardware architecture being used. It can also be considered to belong to the Type 3 category according to the classification of [Crainic & Toulouse, 2003]. There is some flexibility and simplification in presenting Algorithm 5.1 from the point of view of only one agent (or replica). The advantage comes from the fact that all agents behave in the same way on different data, that is, the algorithm takes a data parallel approach.

Line 1 is for the input of the graph G, the number of available colours k, the multiplier to the neighbourhood size M, the number of replicas P, the temperature T and the field strength Γ. Unlike in the initial version of quantum annealing for graph colouring in

Algorithm 4.2 in chapter 4, there is no need for the inverse decrement rate $MaxSteps$ here as Γ is kept constant throughout the duration of the algorithm. Line 2 initializes the estimation of the neighbourhood size. Line 3 announces the discovery of a proper k-colouring, or that no proper k-colouring could be found after the satisfaction of a termination criterion such as a time limit. On line 4, each agent initializes its configuration ω randomly and also randomly initializes place-holder copies of its two neighbouring configurations ω_b and ω_f. On line 5 each agent starts a while loop, which exits when a termination condition such as a time limit is satisfied.

Algorithm 5.1: Messaging protocol for each quantum annealing replica

```
1.  Input: $G = (V, E)$, $k$, $M$,$P$,$T$,$\Gamma$
2.  Initialize: $N_s \leftarrow |V|.k$
3.  Output: “Yes” if a proper $k$-colouring is found or “No” otherwise
4.  Randomly initialize configurations $\omega$, and local neighbour copies $\omega_b$ and $\omega_f$
5.  While termination condition is not met
6.    If distances $D(self, b)$ and $D(self, f)$ are each larger than $|V|/10$
7.      Send own configuration $\omega$ to neighbours $b$ and $f$
8.      Receive and update $\omega_b$ and $\omega_f$ if available
9.    Else
10.     Perturb $\omega$ by randomly changing the colour of a  of $|V|/10$ random vertices
11.   $attempted \leftarrow 0$
12.   While $attempted < M.N_s$
13.     Randomly select a vertex $v$  conflicting in $\omega$
14.     Change colour of $v$ to a new randomly selected colour to derive $\omega'$
15.     $\Delta\mathcal{H}_p \leftarrow \mathcal{H}_p(\omega') - \mathcal{H}_p(\omega)$
16.     Calculate $\Delta\mathcal{H}$ using $\omega_b$ and $\omega_f$
17.     If $\Delta\mathcal{H}_p < 0$ Or $\Delta\mathcal{H} < 0$
18.       $\omega = \omega'$
19.     Else
20.       With probability $\exp(-\Delta\mathcal{H}/T)$, $\omega = \omega'$
21.     $attempted \leftarrow attempted + 1$
22.   End While
23. End While
```

On line 6, each agent $Self$ computes the partition distance between itself and its two neighbours b and f as $D(Self, b)$ and $D(Self, f)$ respectively and checks if each of the distances is larger than $R = |V|/10$. If this is the case, then $Self$ executes line 7 and line 8, sending its configuration to b and f and also receiving their own configurations if they have been sent. Otherwise, this means that $Self$ is too close to at least one of b and f. The agent $Self$ therefore proceeds to line 10 and perturbs its configuration by randomly choosing $R = |V|/10$ vertices and changing the colour of each of them to a randomly chosen colour. On the remaining lines, each agent executes steps similar to those at the corresponding stage in the initial graph colouring quantum annealing algorithm in Algorithm 4.2 in chapter 4. The main compromise in using the data parallel distributed algorithm in Algorithm 5.1 is that each agent has to make use of a slightly outdated state of its neighbours in order to have the increased autonomy and the efficiency gains which follow from not having to synchronize every time a change is made. Experimental results in Section 5.4 show that this compromise does not result in any noticeable drawback.

5.4 Results and Discussion

5.4.1 Experimental set-up

Various aspects of the enhanced graph colouring quantum annealing algorithm were tested on some of the most challenging graphs from the DIMACS benchmarks. The enhanced quantum annealing algorithm leaves Γ fixed, intervenes to control the similarity of the replicas, and is parallelized by an agent-based messaging protocol. Implementation was done in GNU C++ and run on a Linux operating system with hardware consisting of a single desktop PC with 6GB of RAM and a six-core processor with 2.6GHz per core.

An important feature of the algorithm is the parallelism. Algorithm 5.1 was designed to be adaptable to different architectures such as shared memory and message passing, or a combination of both. The current implementation uses Open Multiprocessing (OpenMP) on the shared memory architecture of a multi-core desktop PC. OpenMP is a parallelism standard built into the GNU C++ compiler, which takes care of low level parallelization details after required directives have been specified. In terms of memory management, there are global areas that can safely be accessed by all agents, and there are local areas exclusive to each individual agent.

The most straightforward areas that can be designated global are the read-only data structures for the graph. These consist of the adjacency matrix and the adjacency lists. Other read-only data structures are the ones for the look-up tables for the exponential function. It is clear that no corruption can occur here as the agents do not write to these data structures. The more critical cases involve giving each agent access to its neighbour's configurations in order to calculate changes in spin products and the kinetic energy, corresponding to line 16 of Algorithm 5.1. This is tackled by providing a collection of the copies of each agent's configurations that is maintained as read-only when the agents are searching. There is then a synchronization point corresponding to lines 6-8, where a master thread brings the shared copy of the agents' configurations up to date with the local copies. Examples of data that need to be localized and individualized to each agent include the array storing the colouring configuration, the dynamic set of conflicting vertices, and the list of dynamic sets for the colour classes, which were discussed in Section 4.3 of chapter 4. Additionally any data structures used by the pseudo random number generator should be exclusive to each agent to avoid corruption.

Load balancing conditions are favourable to our implementation as each agent attempts the same $M.N_S$ moves before a synchronization barrier is encountered where the master thread updates the shared copy of configurations. Out of the six processor cores available, only five were used so that the sixth was free for any system processes. In our current experiments the number of replicas was set to $P = 20$, meaning that there were always 20 agents. OpenMP automatically load balances these among the 5 available processors.

The problem instance specific values to which PT and Γ were fixed were obtained by the procedure in Section 4.4.2 of performing short trial runs on easier related instances. In many cases, the tuned parameter values for PT and Γ are the same as those of chapter 4. To take advantage of the additional processing power afforded by the parallelization, the neighbourhood estimation multiplier was set to $M = 8$ in this chapter, as opposed to $M = 4$ in chapter 4.

5.4.2 Experimental results

The implementation of the enhanced quantum annealing from Algorithm 5.1 described in the Section 5.4.1 was run multiple times on instances from some difficult DIMACS graphs. The results are presented in Table 5.1.

Table 5.1: Results for quantum annealing with 5 processor cores

Graph(k^*)[a]	k	PT	Γ	Attempted	Accepted	t[m]	Hit
DSJC500.5(48)	48	0.35	0.70	2.9×10^9	5.8×10^7	1	10/10
DSJC500.9(126)	126	0.20	0.37	6.9×10^9	7.0×10^9	3	10/10
DSJC1000.1(20)	20	0.44	1.05	4.7×10^9	2.3×10^8	3	10/10
DSJC1000.5(83)	83	0.36	0.67	3.9×10^{10}	4.6×10^8	24	10/10
DSJC1000.9(222)	222	0.19	0.375	1.8×10^{11}	8.1×10^8	84	2/10
Le450_25c(25)	25	0.28	0.55	7.0×10^{10}	1.4×10^9	26	10/10
Le450_25d(25)	25	0.28	0.55	1.4×10^{11}	2.3×10^9	49	7/10
Flat300_28_0(28)	31	0.35	0.70	1.5×10^8	4.7×10^6	< 1	10/10
Flat1000_76_0(82)	82	0.36	0.67	4.0×10^{10}	4.7×10^8	24	10/10
R1000.5(234)	238	0.11	0.07	5.1×10^{10}	9.4×10^8	45	4/10
DSJR500.5(122)	122	0.15	0.10	9.6×10^9	3.1×10^8	7	5/10
R250.5(65)	65	0.10	0.10	3.5×10^9	2.1×10^8	2	10/10
Latin_square_10(98)	**97**	0.45	0.95	2.0×10^{11}	2.0×10^9	132	1/10
C2000.5(148)	**147**	0.32	0.60	1.5×10^{12}	7.0×10^9	928	5/5

[a]The values for k^* in the table were the best in the literature before our paper [Titiloye & Crispin, 2011b] was accepted for publication. Some of these were subsequently improved as a result of our further work presented in chapter 6 and published in [Titiloye & Crispin, 2012]. More information about the origin of the k^* values can be found in [Lü & Hao, 2010].

Column 1 (Graph) contains the name of the graph and the number of colours used in the best colourings in the literature. Column 2 (k) contains the number of colours used by the best results obtained in our experiments on each graph. Columns 3 (PT) and 4 (Γ) contain the tuned values of the resultant temperature PT and the field strength Γ respectively for each instance. Columns 5 (Attempted) and 6 (Accepted) report the average number of attempted moves and the average number of accepted moves respectively for each problem instance. Column 7 (t) contains the average wall-clock time in minutes for a successful run, while column 8 (Hit) reports the success rate out of a number of runs. Experiments were run with a termination condition of a limit on the wall-clock time used. This was set to 1.5 hours for all graphs except Latin_square_10 and C2000.5, which were set to 2.5 and 20 hours respectively.

Enhanced quantum annealing finds a new upper bound of 97 on the chromatic number of the Latin square graph Latin_square_10. Additionally the algorithm improves on recent results of 148-colourings for the random graph C2000.5 by finding 147-colourings. The Leighton graphs Le450_25c and Le450_25d are already known to be coloured optimally, and therefore no improvement could have been made there. On the other graphs, exploratory experiments carried out with longer running times suggested that they would not be improved. The results of many of these graph had also not been improved for about two decades despite several attempts by researchers. The robustness of the enhanced algorithm is also better than the initial quantum annealing of chapter 4 in terms of the success ratio (Hit) of the algorithm reported in the last column of Table 5.1.

5.4.3 Comparison with alternative algorithms

In this section, the results from the enhanced quantum annealing algorithm are compared to those from six important publications. In Table 5.2, these are indicated as Mor1996

[Morgenstern, 1996], GHZ2008 [Galinier et al., 2008], MMT2008 [Malaguti et al., 2008], BZ2008 [Blöchliger & Zufferey, 2008], LH2010 [Lü & Hao, 2010], and PHK2010 [Porumbel et al., 2010b]. They have been discussed in Section 4.4.5 of chapter 4. It is customary to compare different graph colouring algorithms using the criterion of the lowest value of k attained as other criteria such as the time taken are difficult to compare in a fair way. Another good reason for using the quality criterion is that it is a good discriminator between excellent algorithms and poorer ones. Most of the time a substandard graph colouring algorithm is not able to replicate the results of better algorithms even when allowed an exorbitant amount of time. Quantum annealing matches all the other algorithms on the random graphs and just like the initial version in chapter 4, finds a 222-colouring for DSJC1000.9, which none of the others had been able to report. Our algorithm finds 25-colourings for the two challenging Leighton graphs Le450_25c and Le450_25d with high success ratios of 100% and 70%.

Table 5.2: A comparison between quantum annealing and some leading algorithms

Graph(k^*)[a]	QA	Mor 1996	GHZ 2008	MMT 2008	BZ 2008	LH 2010	PHK 2010
DSJC500.5(48)	48	48	48	48	48	48	48
DSJC500.9(126)	126	126	126	127	126	126	126
DSJC1000.1(20)	20	21	20	20	20	20	20
DSJC1000.5(83)	83	84	84	83	89	83	83
DSJC1000.9(222)	222	226	224	224	225	223	223
Le450_25c(25)	25	25	26	25	25	25	25
Le450_25d(25)	25	25	26	25	25	25	25
Flat300_28_0(28)	31	31	31	31	28	29	29
Flat1000_76_0(82)	82	84	84	82	87	82	82
R1000.5(234)	238	241	-	234	247	245	237
DSJR500.5(122)	122	123	125	122	125	122	122
R250.5(65)	65	65	-	65	65	65	65
Latin_square_10(98)	97	98	104	101	-	99	98
C2000.5(148)	147	-	-	-	-	148	148

[a]See the footnote of Table 5.1 with regards to k^* values

Many of the algorithms in the literature not listed here struggle with finding 25-colourings for these Leighton graphs. The graph Flat300_28_0 is a peculiar case that was not coloured with

28-colours until 2008 by [Blöchliger & Zufferey, 2008] despite being used in experiments in the literature since the 1993 DIMACS competition. Although we report only 31-colourings in our general experiments, we show in Section 5.4.4 how special cases of our algorithm can find 28-colourings. On the larger flat graph Flat1000_76_0, quantum annealing matches the 82-colourings first reported by [Malaguti et al., 2008], and subsequently reported by [Lü & Hao, 2010] and [Porumbel et al., 2010b].

Quantum annealing is also strong on the random geometric graphs R250.5 and DSJR500.5, reaching the best ever results of 65 and 122 respectively. The algorithms by [Morgenstern, 1996] and [Blöchliger & Zufferey, 2008] could only reach 123 and 125 respectively.

The 97-colouring for the Latin square graph Latin_square_10 is new, and an improvement over the previous best results of 98-colourings first reported by [Morgenstern, 1996], and later by [Porumbel et al., 2010b]. The 147-colourings for the large random graph C2000.5 outperform the result of 148-colourings reported by both [Lü & Hao, 2010] and [Porumbel et al., 2010b].

5.4.4 The influence of introduced features

It is important to measure the effectiveness of our parallel algorithm at providing a speedup when running on multiple processors instead of just one. Two widely used measurements in the parallelization of algorithms are termed speedup and efficiency. Adopting the notation from [Cung et al., 2002], for a problem $\mathcal{P}$, and a parallel algorithm $\mathcal{A}_{\mathcal{P}}$ running on a machine $\mathcal{M}$ with q identical processors, then $T_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(p)$ is the time taken by $\mathcal{A}_{\mathcal{P}}$ to solve $\mathcal{P}$ on $\mathcal{M}$ using $p \leq q$ processors. Additionally, $T_{\mathcal{A}_s}$ represents the time taken by the best sequential algorithm running on the same type of processor in $\mathcal{M}$. With these definitions, the speedup $s_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(p)$ provided with p processors can be written as

$$s_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(p) = \frac{T_{\mathcal{A}_S}}{T_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(p)}\,.$$

In the context of our experiments and as mentioned in [Cung et al., 2002], $T_{\mathcal{A}_s}$ is replaced by $T_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(1)$, the time it takes to run our parallel algorithm using only one processor of $\mathcal{M}$. In order to present numerical results for the speedup, the instance ($G =$ DSJC500.5, $k = 48$) was solved a hundred times first with the parallel algorithm, initially with 1 processor core and then with 5 processor cores. The average time of 285 seconds was used with 1 processor, while this was 70 seconds with 5 processors. The ratio $s_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(p)$ for the given problem instance is then $285/70$, yielding a value of about 4. This means that using 5 processors results in a speedup of about 4.

The second measurement denoted efficiency is a generalisation of the speedup that aims to factor out the number of processors. The efficiency $\eta_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(p)$ is given as

$$\eta_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(p) = \frac{s_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(p)}{p}\,.$$

The closer the value of the efficiency is to 1, the better the effect of the parallelization. An efficiency of 1 represents a linear speedup which is generally difficult to achieve. In the case of our example with the problem instance ($G =$ DSJC500.5, $k = 48$), the efficiency is given by $\eta_{\mathcal{A}_{\mathcal{P}},\mathcal{M}}(p) = 4/5$ or 0.8.

The benefit of the enhancement of not having to decrease the field strength Γ is clear in that the results of Table 5.1 are at least as good as (and in some cases better than) the ones in Table 4.2 in chapter 4, without the burden of having to specify an extra parameter $MaxSteps$ to control the rate of change of Γ. In addition to having to find an appropriate initial value for Γ, the initial implementation of quantum annealing ran the risk that an inappropriate value of $MaxSteps$ would be set, causing failure. This risk was removed with the fixed-Γ enhancement and efforts only need to be concentrated at finding a suitable value of Γ and T for each instance.

Additional experiments were carried out to demonstrate the efficacy of the replica spacing enhancement. The implementation of Algorithm 5.1 was run 10 times for the instance $(G = \text{DSJC1000.5},\ k = 83)$ with replica spacing disabled and with the parameter settings $PT = 0.31$ and $\Gamma = 0.67$. Apart from the disabling of the replica spacing component, all other conditions were the same as those used for the results in Table 5.1, including the 5-hour time limit. None of the 10 attempts were successful at reaching an 83-colouring. This was in contrast to the 100% success rate achieved with the replica spacing component in place. It was only when the larger value of 0.72 was set for Γ and the time limit was relaxed that solutions were found. However, this led to a three-fold increase in the time taken. Generally, if all other conditions are equal, a simulation with a higher value for Γ would take longer to complete, at it spends a lot of time in the regions where the moves made are similar to a random walk. It can be deduced that replica spacing allows a lower value of Γ to be set with little risk of premature convergence, while simulations not using replica spacing may be forced to use higher values of Γ, and hence larger amounts of time in order solve the same problem instance.

Table 5.3 displays the number of searches by the replicas and the number of times a perturbation had to be applied to a configuration because it was too close to its connected neighbours. As can be seen from Table 5.3, the perturbation count is usually only a small fraction of the total number of searches. But this has usually been enough to prevent premature convergence of the replicas. The figures in Table 5.3 are averaged over the number of successful runs in the same experiments as in Table 5.1

Table 5.3: Measuring the amount of perturbation used

Graph	k	Replica search count	Perturbation count
DSJC500.5	48	14880	0
DSJC1000.1	20	29313	51
DSJC1000.5	83	58860	1025
Le450_25c	25	330706	23
Le450_25d	25	1553078	32
Flat100_76_0	82	60358	1202
C2000.5	147	668190	1150

The geometric random graphs never produced configurations that caused replicas to come with $|V|/10$ of each other. Therefore the replica spacing routine was never triggered for these graphs. The very dense random graphs with $p = 0.9$ such as DSJC500.9 are also not very reliant on perturbations and the replica spacing component. Our experiments showed that even when two replicas came within $|V|/10$ of each other, this usually corrected itself and the replicas moved away from each other again. Turning on the replica spacing component initially appeared to be disruptive to these graphs and was therefore not used for the experiments in [Titiloye & Crispin, 2011b]. However further experiments revealed that although the replica spacing component could cause very dense random graphs to take a longer time to produce colourings, the process itself is not disrupted and robustness can actually be improved by using it. We therefore switched replica spacing on for very dense graphs in further experiments related to the study of parameter tuning in chapter 6.

5.4.5 Variations and special cases

One feature our Monte Carlo quantum annealing algorithm shares with Evolutionary algorithms is that they are both based on the interaction between individuals in a population. In the case of graph colouring, they work on the premise that two different configurations of high quality often share common components. There is usually a correlation between low lying local minima in terms of the composition of their colour classes [Culberson & Gent, 2001; Hamiez & Hao, 2004]. However we should expect the level of this correlation to be dependent on the probability distribution from which a graph is drawn, or the method by which it is constructed. When dealing with a graph exhibiting a low level of correlation between its colouring local minima, forcing a high amount of cooperation between individuals may not yield much benefit, and may even be counterproductive.

Quantum annealing does have a method to finely regulate the amount of cooperation in the replicas by adjusting the initial value set for the field strength Γ. A high value for Γ means less interaction and more individual freedom, while a low level for Γ means more interaction and less individual freedom. One graph from the DIMACS benchmarks that has traditionally not been able to benefit substantially from the interaction of a population is the Latin square graph Latin_square_10. Some key results from leading algorithms seem to suggest that colouring Latin_square_10 with just the local search component is more productive than using the full Evolutionary-Tabu approach. For example, the Evolutionary-Tabu algorithm in [Malaguti et al., 2008] could not match the long-standing upper bound of 98 found by [Morgenstern et al., 1996], and could only colour this graph with 101 colours. (From the results of quantum annealing in Table 5.1, we now know that Latin_square_10 is in fact 97-colourable). Another leading Evolutionary-Tabu algorithm by [Lü & Hao, 2010] only found 99-colourings. In a third Evolutionary-Tabu publication by [Porumbel et al., 2010b], the regular approach which works well for other graphs only finds 100-colourings. However a variant of the main approach does find 98-colourings. Interestingly, the 98-colourings were found by increasing the number of iterations used by the Tabu search phase a hundred fold from 100,000 to 10,000,000. Although it was not stated explicitly in [Porumbel et al., 2010b], this minimizes the effect of the evolutionary phase. Even then, the success rate was $4/30$ with an average time of 12 hours on a single processor core. In contrast, as can be seen from Table 4.2 in chapter 4, quantum annealing achieves a 100% success rate in finding 98-colourings, while requiring less than 1 hour on a single processor core.

In order to confirm that quantum annealing benefits from less cooperation for Latin_square_10, the field strength was set to a very large value of $\Gamma = 10^9$, which decouples the replicas, making them behave like parallel classical annealing. The value of the temperature was kept the same at $PT = 0.45$. There was no change in the effectiveness of our algorithm in finding 98-colourings. In contrast, trying to find 83-colourings for the random graph DSJC1000.5 by setting the same $\Gamma = 10^9$ failed. The replicas simply behaved in a manner similar to a random walk. The next stage was to determine if the success rate for 97-colourings could be improved from the 10% reported in Table 5.1 by decoupling the replicas. This was indeed the case. The success rate turned out to be 100% in 5 runs in an average wall-clock time of about 20 hours with 5 processor cores. It was checked that allowing this much time while having interacting replicas with a low Γ did not have the same effect. Decoupling the replicas was the key to getting an improved performance for Latin_square_10.

Another graph that can be considered an anomaly is the flat graph Flat300_28_0. Despite consisting of only 300 vertices and studied in the context of graph colouring since 1993, it was only in 2008 that 28 colourings were found [Blöchliger & Zufferey, 2008]. Before then, no results better than 31-colourings were reported in the literature. Here again, non-population based metaheuristics have fared better than population based ones. In particular it was noticed by [Blöchliger & Zufferey, 2008] that the impasse-class neighbourhood [Morgenstern, 1996] was advantageous for the graph. Since the kinetic energy is not important in a decoupled version of quantum annealing, a variant of our algorithm that used the impasse-class neighbourhood could be readily implemented. This variant was able to find 28-colourings in an average time of less than 1 hour on 5 processors.

We next reverted back to the basic fixed-k neighbourhood used for all other instances and found that if additional time of up to 24 hours on 5 processors was allowed, 28-colourings could be found using the usual neighbourhood as well, with a 100% success ratio over 5 independent runs.

5.5 Conclusion

The main aim of this chapter was to provide enhancements to the quantum annealing algorithm in order to improve its performance for the graph colouring problem. The first change presented was a fixed-Γ approach inspired by analogous fixed-temperature schedules for classical annealing. This modification led to a simplification of the tuning process and an increase in the robustness of the quantum annealing algorithm. The second enhancement involved the spacing of replicas. This was inspired by similar techniques in evolutionary algorithms designed to prevent premature convergence. Experiments showed that this improved the results of quantum annealing in terms of the success rates, and it enabled a new result to be found for a large random graph. The third enhancement was the parallelization of the algorithm by the presentation of an agent based messaging protocol. Experiments showed that an impressive speedup was achieved, enabling hard problem instances to be solved in much quicker times than would otherwise have been possible. Variations of quantum annealing based on the exploration of a different neighbourhood were also considered, enabling an atypical problem instance to be optimally solved quickly.

Chapter 6

Tuning Patterns for the Colouring of Random Graphs

This chapter deals with investigations into patterns that help in the decision of how to set the parameters of quantum annealing in order to increase the chances of finding a global minimum for the k-colouring problem when random graphs are under consideration.

Section 6.1 and Section 6.2 review the existing approaches to tuning classical annealing and quantum annealing respectively. Section 6.3 addresses tuning features specific to graph colouring for quantum annealing, with an emphasis on random graphs, which are widely used as benchmark problems. In Section 6.4, we report the first 82-colourings for DSJC1000.5 in the literature, discovered after observing variances for parameter tuning of a random graph as the value of k changes. Section 6.5 investigates the evolution of the acceptance ratio for favourable and unfavourable parameter settings and important patterns are observed. This contributed to the discovery of additional new benchmark results, including 47-colourings for DSJC500.5, 145-colourings for C2000.5, 259-colourings for C4000.5 and 400-colourings for C2000.9.

6.1 Approaches to Tuning Classical Annealing

Standard versions of the classical annealing algorithm require the specification of a temperature parameter T, and a rate of decrease of T. The setting of these parameters is often termed an annealing schedule. In order to guarantee convergence to a global minimum of any cost function or Hamiltonian $\mathcal{H}_p$, an inverse logarithmic schedule is needed. From the theory of Markov chains, it is known that such a schedule requires that the temperature at each Monte Carlo step satisfies the relation $T \geq T_0/(1 + \log(1 + t))$ where t is a measure of time in Monte Carlo steps and T_0, the initial temperature, is a large constant dependent on the characteristics of the problem at hand [Geman & Geman, 1984; Hajek, 1985; Mitra et al., 1986]. A bound for the value of the constant T_0 was given in [Hajek, 1985] as $T_0 \geq d^*$, where d^* is maximum value of the depth d of all local minima that do not correspond to a global minimum. The depth d of a locally minimal configuration is the smallest value of an uphill move $(\Delta\mathcal{H}_p > 0)$ needed in order to escape from that local minimum. This is more rigorously defined in [Hajek, 1985].

Although the inverse logarithmic schedule (also known as the canonical schedule) guarantees convergence to the global minimum, it is extremely slow. As a result, faster schedules that work well in practice but may forgo guaranteed convergence have to be used. Annealing schedules are categorized into static and adaptive approaches. In the static approach, an initial temperature and rate of decrease of the temperature are specified, and these never change until the algorithm is terminated. On the other hand, the adaptive approach such as the one by [Lam & Delosme, 1988] attempts to react to information obtained during the search process, and to use this to determine what the temperature or the rate of decrease should be at any given time. Two main types of static schedules are the geometric and linear schedules. The geometric schedule, also known as the exponential schedule was first used in [Kirkpatrick et al., 1983].

A geometric schedule is characterized by the relation $T_t = \alpha^t T_0$. Here again, T_0 is an initial temperature larger than zero and T_t is the temperature at time t. The constant α is chosen to be between zero and one. For example, α was set to 0.9 in [Kirkpatrick et al., 1983]. At each temperature T_t, a Markov chain consisting of several moves is searched before the temperature is reduced in the next time step. One method of tuning T_0 is to calculate the maximum value of $\Delta\mathcal{H}_p$, the change in the cost function, and set T_0 to it. This approach is known to be computationally intensive [Aarts et al., 1997]. Instead, test runs can be used to empirically determine a value for T_0 at which all moves or almost all moves are accepted [Kirkpatrick et al., 1983].

A linear schedule can be defined as $T_t = T_0(1 - t/t_{max})$, where t_{max} is a pre-defined maximum number of Monte Carlo steps (equivalent to $MaxSteps$ in chapter 4) for which we are willing to run the algorithm. It is simpler than the geometric schedule. Although the linear schedule is not as widely used as the geometric schedule, an analytical study has determined with a problem of small size that both schedules are of similar effectiveness [Strenski & Kirkpatrick, 1991]. The linear temperature schedule was used in the implementation of classical annealing that was compared to quantum annealing for the travelling salesman problem [Martoňák et al., 2004], and also for the classical annealing graph colouring algorithm in chapter 4 of this thesis. Although T_0 can be tuned as already described for the geometric schedule, a lower starting temperature obtained by exploiting problem specific characteristics may be more effective in some circumstances. This has also been observed for a rarely used schedule that maintains such a heuristically obtained low temperature fixed T_0 for the whole duration of the algorithm [Connolly, 1990; Cohn & Fielding, 1999; Fielding, 2000]. In [Connolly, 1990], T_0 was determined for the quadratic assignment problem by performing short trial runs and choosing the temperature according to the lowest and highest values for the cost function during the trial. Instead of starting from a random initial configuration, a construction heuristic could be used to provide a better initial configuration which could permit the temperature to be tuned to a low value without the algorithm getting trapped [Johnson & McGeoch, 1997].

.

6.2 Approaches to Tuning Quantum Annealing

In a quantum annealing algorithm, the field strength Γ is the main tuning parameter. However, many of the same issues still apply, and it is possible to try the same annealing schedules used for classical annealing. The number of replicas P needs to be set for quantum annealing. It was previously unclear how sensitive the value of P was over a variety of problem instances from the same optimization problem. This was partly because of a shortcoming in the Physics quantum annealing literature, whereby it was customary to consider only one problem instance in all experiments for a publication. For example, only the one TSPLIB [Reinelt, 1991] instance pr1002 was used in a study of quantum annealing for the travelling salesman problem [Martoňák et al., 2004]. Additionally only a single 10,000 variable random 3-SAT instance was used in an application of quantum annealing to Boolean satisfiability [Battaglia et al., 2005]. However, chapter 4 of this thesis considered several problem instances for the graph k-colouring problem using the same value $P = 10$ for all of them and achieving competitive results. Therefore it would appear that good values of P do not vary considerably from one instance to another, and one can concentrate on finding the patterns that determine good initial values of the field strength Γ and the temperature T for various problem instances, and at what rate they should be decremented if at all. It is our aim to use the simplest annealing schedule that can be effective, as an overly complicated tuning procedure threatens the usability of an algorithm. Standard versions of quantum annealing prescribe that the temperature be fixed (to a determined low value), and the field strength set to a high value to be gradually lowered towards zero [Santoro et al., 2002]. There are variants of quantum annealing that decrement the temperature as well as the field strength [Lee & Berne, 2000; Kurihara et al., 2009; Sato et al., 2009]. It is possible that the best treatment of the temperature depends on the optimization problem at hand. A fixed temperature appears to be favourable for the graph k-colouring problem just as in past studies for the Ising model [Santoro et al., 2002; Martoňák et al., 2002] and the travelling salesman problem [Martoňák et al., 2004]. Suitable values of the fixed temperature still have to be sought.

At least partially because classical annealing has been studied for a lot longer and a lot more than quantum annealing, knowledge about parameter tuning for quantum annealing is not as advanced as that of classical annealing. The process of determining the starting values for Γ and T for the two-dimensional random Ising Model study in [Santoro et al., 2002] was not very systematic. Instead, a number of trial runs were performed and the best performing values were chosen. The study in [Martoňák et al., 2002] attempted to go further by examining the dependence of quantum annealing on varying values for the number of replicas P and a low temperature T. The field strength Γ was determined in the same way as in [Santoro et al., 2002]. In both cases, Γ was then slowly lowered towards zero with a linear annealing schedule. During several experiments on the same random Ising Model problem instance, three different values of the quantity PT were tried. Values for the resultant temperature of $PT = 1$, $PT = 1.5$ and $PT = 2$ were experimented with, while also varying P from between 5 and 50 in various experiments. Various plots were then produced to show the relationship between the parameter settings and the rate of decrease of the cost function over different lengths of annealing time. This way, it was determined that the settings $P = 40$ and $PT = 1$ were very good for the two dimensional random Ising Model considered.

In the case of the quantum annealing study for the travelling salesman problem in [Martoňák et al., 2002], quantum annealing was tuned by first tuning classical annealing. While classical annealing can operate from a high temperature that is gradually lowered, the standard quantum annealing needs a low fixed temperature. In order to derive a suitable value for the fixed temperature to be used for an instance pr1002, several short classical annealing trial annealing runs of various lengths were performed. A dynamical temperature T_{dyn} defined as the temperature for which cooling curves for various lengths of annealing times started to differ was determined. The initial temperature for longer classical annealing runs was then set as $T_0 = T_{dyn}$ for an alternative type of schedule with low starting temperatures. More interestingly, it was observed that a suitable way to tune quantum annealing was to set $PT = T_{dyn}$. The performance of quantum annealing is less sensitive to the value of P, and was set as $P = 30$ based on the availability of computing resources. To set higher values for P, it is sufficient to simply adjust T accordingly such that the same working value for PT is maintained. This behaviour was also observed for the graph k-colouring problem in chapter 4 and chapter 5. Setting the value of the field strength Γ for the travelling salesman problem appeared to require less care, and a large initial value of $\Gamma_0 = 300$ was set and linearly reduced towards zero.

6.3 Tuning for Graph Colouring

We now focus on parameter tuning for the graph k-colouring problem. The nature of the k-colouring problem presents challenges as well as opportunities not generally present in other combinatorial optimization problems. For any graph G and number of colours k, the k-colouring problem is concerned with whether the instance (G, k) possesses a configuration ω such that the number of conflicts (or cost function) $\mathcal{H}_p$ is zero. Although it has become customary for researchers to test their algorithms on problem instances with values of k already known to be solvable in the literature, it is also possible to perform a higher level optimization procedure as follows: First colour G using a greedy heuristic [Brélaz, 1979; Leighton, 1979] to obtain an upper bound UB for the chromatic number, then set $k = UB - 1$

and attempt to find a colouring with the main solver. If successful, increasingly difficult instances $(G, k-1)$, $(G, k-2)$ and so on are attempted, until some readily computable lower bound such as a clique size in G is reached, or another termination condition such as a time limit. Something similar to the described approach was used in [Malaguti et al., 2008]. Apart from being a higher level optimization procedure to estimate the chromatic number χ of a graph, it forms the basis of a useful tuning procedure. Since there are a series of related problem instances of progressively increasing difficulty, the same (or a minor adjustment) of the parameters used for (G, k) can be tried for $(G, k-1)$. Carrying out most of the tuning on the easier instances with a larger k can result in an easier tuning procedure. This is what was done in chapter 4 and chapter 5. More specifically it was observed that in many cases, the same low fixed temperature T, or the same resultant temperature PT could be used for instance (G, k) and $(G, k-1)$. Additionally, once a suitable value for the field strength Γ had been found for (G, k), a reasonable value of Γ for $(G, k-1)$ was usually obtainable by adjusting that of (G, k) by a few hundredths in a somewhat predictable manner.

The structure of a graph influences what parameters are likely to be suitable. Invariants like the edge density, maximum degree, minimum degree and average degree affect the energy landscape induced by a problem instance. Intuitively, different graphs that are generated from the same probability distribution or with a similar strategy are likely to share a common tuning pattern that can be discovered. In order to verify that this is indeed the case, and to study what patterns there are if any, it is useful to focus on graphs generated from the same distribution. The Erdös-Rényi [Gilbert, 1959; Erdös & Rényi, 1959; Erdös & Rényi, 1960; Bollobás, 2001] is a well-known model of random graphs studied extensively theoretically, and widely used in experiments for various graph optimization problems. They are easy to generate, and a standard benchmark of graphs of various sizes and densities provided by the second DIMACS [Johnson and Trick, 1996] competition are readily available for researchers to compare results. In the Erdös-Rényi $G(n,p)$ model, a graph of n vertices is generated by independently including each of the $\binom{n}{2}$ possible edges with probability p. The case of $p = 0.5$ is especially appealing as every graph from that distribution is chosen uniformly at random from all the $2^{\binom{n}{2}}$ graphs with n vertices. Additionally many asymptotical properties of the $G(n,p)$ model are derived from the case of $p = 0.5$ as $n \to \infty$ [Bollobás, 2001]. Although many of the characteristics of the $G(n,p)$ model such as connectivity, maximum clique size and chromatic number are customarily studied in an asymptotical context, some attention has been devoted to the situations where the number of vertices n is only in the region of hundreds or a few thousands [Bollobás & Thomason, 1985]. It is noteworthy that the graph k-colouring problem is still hard when restricted to Erdös-Rényi random graphs if instances are chosen near the phase transition were k is very close or equal to the chromatic number χ [Achlioptas et al., 2005; Zdeborová & Krząkała, 2007]. Particularly, no known polynomial time algorithm is guaranteed to colour Erdös-Rényi random graphs with $(2-\varepsilon)\chi$ colours where $\varepsilon > 0$ is fixed [Achlioptas & Coja-Oghlan, 2008]. It is known that the move acceptance ratio during annealing is strongly related to parameter tuning [Lam & Delosme, 1988]. In the next section we exploit a behaviour exhibited by the evolution of the acceptance ratio while colouring random graphs with an annealing algorithm, in order to find good tuning parameters for very low values of k that are close to the chromatic number χ.

6.4 Parameter Tuning Variance for Random Graphs

The Erdös-Rényi $G(n,p)$ random graphs in the DIMACS benchmarks have p ranging between 0.1 and 0.9, and can all be considered dense in comparison to very sparse 3, 4 and 5-

colourable graphs that were considered in [Zdeborová & Krząkała, 2007]. Of particular importance is DSJC1000.5, the only $G(1000, 0.5)$ graph in the DIMACS benchmarks collection. Erdös-Rényi graphs with $n = 1000$ and $p = 0.5$ have an interesting history over the past several decades of graph colouring research. One reason for this is that they are complex enough to be challenging and approachable by a variety of methods, while not being so large that they could not fit easily into the available memory of legacy computers [Bollobás & Thomason, 1985]. Initially, only poor estimates of χ of about 116 for the DSATUR [Brélaz, 1979] greedy heuristic and about 107 for the RLF [Leighton, 1979] greedy heuristic were available [Johnson et al., 1991].

Colourings with an average chromatic number of 95.9 were presented in [Johri & Matula, 1982]. There also, a probabilistic estimation $\overline{\chi}$ of the expected chromatic number of graphs from the $G(1000, 0.5)$ family was given as $\overline{\chi} = 85$. An algorithm presented in [Bollobás & Thomason, 1985] was able to improve on [Johri & Matula, 1982] by providing colourings using an average of 86.5 colours. This family of graphs was also studied early on with Tabu search in [Hertz & Werra, 1987], finding solutions with 87 colours. The research in [Johnson et al., 1991] produced solutions with an average of 85.5 colours on a sample of four $G(1000, 0.5)$ graphs using a semi-exhaustive version of RLF called XRLF. It was here that the now widely used instance DSJC1000.5 was introduced. Solutions using 86 colours were provided, and it was noted that its density of 0.5000152 was slightly higher than those for which solutions with 85 colours were found. Subsequently, there was an improved result by [Morgenstern, 1996] with the provision of solutions using 84 colours using a classical annealing algorithm that was initialized with XRLF. After a few years, 83 colourings for DSJC1000.5 were presented in [Galinier & Hao, 1999]. The upper bound of 83 remained best until the commencement of this thesis, and had been reproduced by several state of the art graph colouring algorithms [Malaguti et al., 2008; Lü & Hao, 2010; Porumbel et al., 2010b].

A powerful probabilistic estimating procedure in [Bollobás & Thomason, 1985] suggested that $\chi \geq 80$ for instances of the $G(1000, 0.5)$ family, and that the lower bound was tight with very high probability. Therefore one of the goals of this thesis was to produce the first 82-colourings of DSJC1000.5. By exploiting a tuning pattern related to the acceptance ratio, we succeeded in attaining this result which had resisted solution since the year 1999.

Using the enhanced version of quantum annealing for graph colouring presented in chapter 5, and setting the number of replicas to $P = 10$, we easily found colourings in the range $87 \leq k \leq 83$ for DSJC1000.5. The hardware setup for the experiments in this chapter consisted of a cluster of two identical desktop personal computers, each with six-core processors. The specifications of the desktop computers were the same as the ones given in chapter 5. The number of replicas was set to $P = 10$ in an attempt to save time during the tackling of the bigger graphs. One processor core was left free on each desktop, and the replicas were shared equally between them. Intra-communication on each desktop was done with OpenMP as in chapter 5, while inter-communication was done with MPI (Message Passing Interface). The resultant temperature of $PT = 0.36$ was used for all values of k, just as in chapter 4 and chapter 5, with increasing values for a fixed field strength Γ in the range between 0.55 and 0.68. The acceptance ratio during annealing can be calculated as the number of accepted moves divided by the number of attempted moves. The resultant quantum annealing temperature of $PT = 0.36$ can be considered quite low as it produces a starting acceptance ratio of about 1.3% for $k = 83$, in comparison to the 50% starting acceptance ratio for classical annealing in [Johnson et al., 1991]. It is notable that a resultant temperature of about $PT = 500$ would have had to be set in order to produce a 50% acceptance ratio for quantum annealing. However the idea of quantum annealing is to control the simulation mainly by the field strength in the presence of a low temperature. Ideally, the temperature would be set to absolute zero so that all optimization is carried out entirely by quantum fluctuations. But the path integral Monte Carlo quantum annealing which our algorithm is based on cannot be simulated at zero temperature [Battaglia et al., 2005].

We observed that setting the resultant temperature to be even slightly less than $PT = 0.36$ for the instance DSJC1000.5 caused our algorithm to take orders of magnitude of additional time to solve $87 \leq k \leq 83$. This was the motivation for keeping the resultant temperature at $PT = 0.36$ for all values of k for DSJC1000.5 so far. It was also observed that the extra difficulty of successively lower values of k could be compensated for by using a slightly larger value for Γ. Specifically with the graph DSJC1000.5, for the values of $k = 87, 86, 85, 84, 83$, the values of the field strength were set to $\Gamma = 0.55, 0.58, 0.61, 0.64$,0.68 respectively. It was observed in chapter 4 and chapter 5 that these increments to Γ are important as the values of k gets lower and more difficult, in order to prevent quantum annealing from getting trapped due to the replicas becoming too similar too quickly.

The first attempt we carried out in seeking valid solutions for the instance with $k = 82$ was to continue the pattern of setting an increased value of Γ while maintaining the resultant temperature at $PT = 0.36$. However, this idea met with obstacles, as setting $\Gamma \geq 0.72$ was too high, causing the replicas to decouple and individualize, while a value of $\Gamma = 0.7$ appeared to be too low, causing simulations to collapse too quickly with very similar replicas which got trapped. Setting various values for Γ between 0.7 and 0.72 did not help either, and several unsuccessful experiments were carried out, each running for up to several days. It became clear that a change in strategy was needed. Even though setting temperatures to below $PT = 0.36$ had been observed to cause unnecessarily long running times for the instances of DSJC1000.5 in the range $87 \leq k \leq 83$, we carried out a series of experiments for $k = 82$ with a reduced resultant temperature of $PT = 0.31$ and set the field strength to $\Gamma = 0.7$. Remarkably, experiments produced a 100% success rate over ten runs with the new parameter settings, thus marking the first time that 82-colourings have ever been found for DSJC1000.5 in the literature [Titiloye & Crispin, 2012]. The successful parameter values of DSJC1000.5 for the range $87 \le k \le 82$ are plotted in Fig 6.1.

It was important to not terminate the algorithm too soon and allow a large enough amount of time as the instance with $k = 82$ took a lot longer to solve than that with $k = 83$. Specifically, it took quantum annealing an average of 5.6×10^{11} attempted moves and 4.4×10^{9} moves for each of the ten successful runs for the problem instance of finding 82-colourings for DSJC1000.5. In contrast, 83-colourings were found with the settings of $PT = 0.36$ after an average of 3.9×10^{10} attempted moves and 4.6×10^{8} moves in chapter 5.

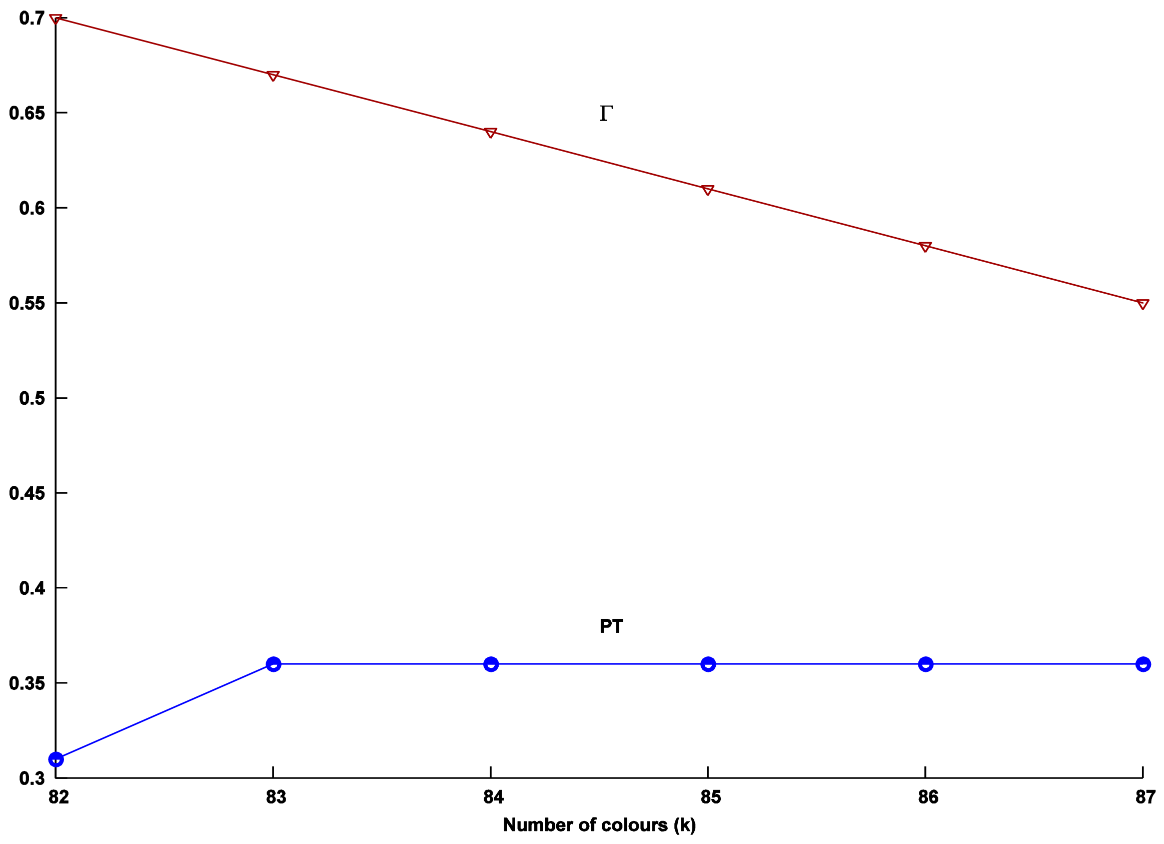

Figure 6.1: Parameter tuning variance with the number of colours for the graph DSJC1000.5. The field strength Γ (red), and the effective temperature *PT* (blue), plotted against the number of available colours *k*.

6.5 Patterns in the Evolution of the Acceptance Ratio

During the experiments in which 82-colourings of DSJC1000.5 were attempted and eventually attained, interesting features of the acceptance ratio were observed. Figure 6.2 shows a typical plot for an unsuccessful experiment with the instance (G = DSJC1000.5, k = 82) using the parameter settings $PT = 0.36$ and $\Gamma = 0.7$. The acceptance ratio first declines over the first few tens of thousands of Monte Carlo steps and then stagnates.

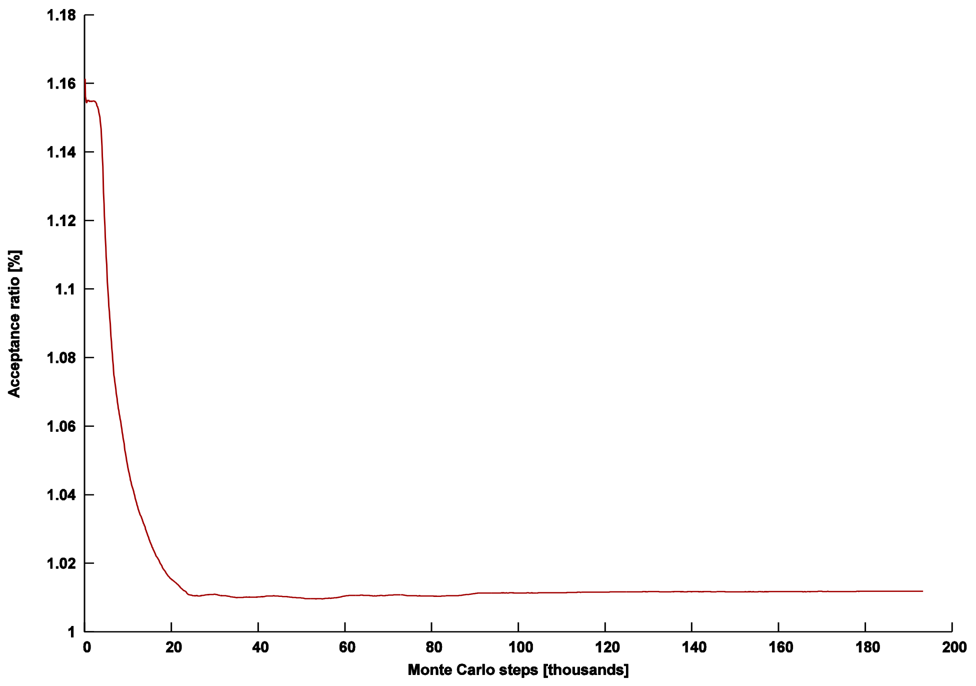

Figure 6.2: An unsuccessful simulation with a declining and stagnating acceptance ratio. The acceptance ratio is plotted against Monte Carlo steps for (G = DSJC1000.5, k = 82) with parameters $\Gamma = 0.7$ and $PT = 0.36$.

In contrast, successful experiments for the same instance with all conditions remaining the same except for the change to $PT = 0.31$ exhibits an unusual pattern of a steadily increasing acceptance ratio over time in Figure 6.3.

It is notable that the declining pattern of the acceptance ratio displayed in Fig 6.2 is usual and generally helpful for easier instances. For instance, solutions for DSJC1000.5 instances for the range $87 \leq k \leq 83$ exhibit a similar declining pattern with $PT = 0.36$ without affecting the success of the experiments. Past classical annealing experiments for k-colouring in [Johnson et al., 1991] also follow a declining pattern in the acceptance ratio. Although in their case it was more obvious that this should happen since the temperature was being decreased over time. Interestingly, when we set $PT = 0.31$ for instances in the $87 \leq k \leq 83$ range, the pattern of the rising acceptance ratio appeared, but solutions were obtained with up to a fivefold increase in computational time. This suggests that $PT = 0.31$ is unnecessary and even inappropriate for the easier instances as this causes quantum annealing to be uncompetitive with the leading algorithms in terms of computational requirements. It is only when the value of k is low and probably close to the chromatic number that the strategy with the higher temperature fails. Only then is it necessary to employ a value of the temperature low enough to induce the increasing acceptance ratio. The patterns exhibited by the acceptance ratio were not unduly influenced by the perturbation from the replica spacing technique introduced in chapter 5 as the patterns were established well before any replica got to within $|V|/10$ of its neighbours.

The pattern of the continuously increasing acceptance ratio can be induced to improve the results for other random graphs in cases where the alternative declining pattern has stopped working.

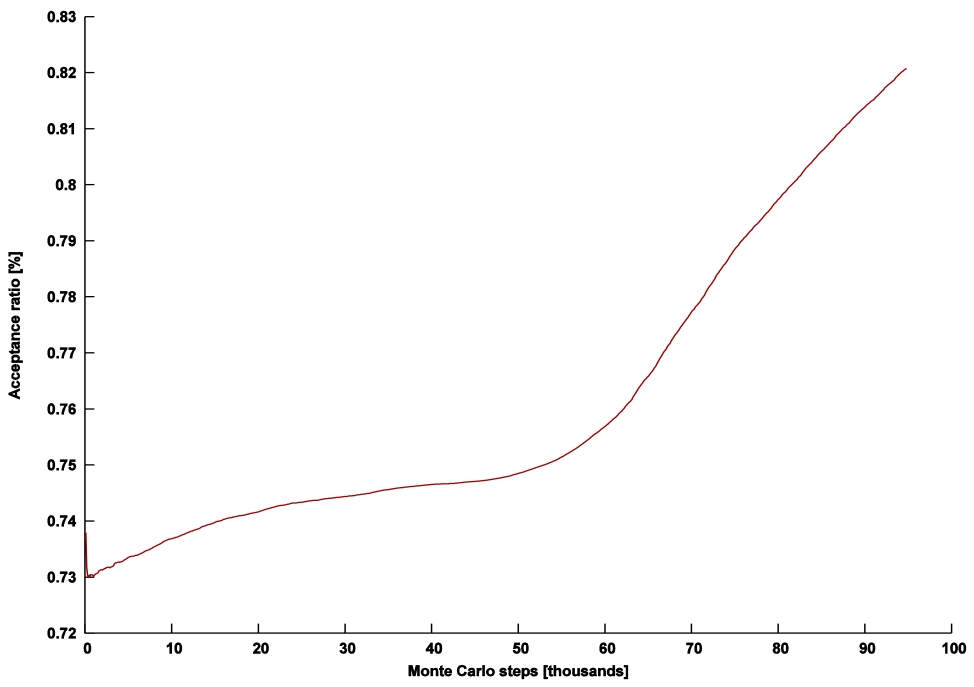

Figure 6.3: A successful simulation with a continuously rising acceptance ratio. The acceptance ratio is plotted against Monte Carlo steps for (G = DSJC1000.5, k = 82) with parameters Γ = 0.7 and PT = 0.31.

Table 6.1 and Table 6.2 show quantum annealing results. These include the discovery of 47-colourings for DSJC500.5, a $G(500, 0.5)$ graph from the DIMACS benchmarks. This is the first improvement in the results of DSJC500.5 since 48-colourings were first discovered by [Morgenstern, 1996]. The easier instance of (G = DSJC500.5, k = 48) was solved in chapter 4 and chapter 5 with the parameter settings $PT = 0.35$ and $\Gamma = 0.70$, which produced a declining acceptance ratio.

Table 6.1: Quantum annealing (QA) colouring results compared with the best alternatives

Graph	QA	Evolutionary-Tabu	Extraction pre-processing
DSJC500.5	**47**	48 [Galinier & Hao, 1999]	48 [Morgenstern, 1996]
DSJC1000.5	**82**	83 [Galinier & Hao, 1999]	83 [Wu & Hao, 2012]
DSJC1000.9	**222**	223 [Lü & Hao, 2010]	222 [Wu & Hao, 2012]
C2000.5	**145**	148 [Lü & Hao, 2010]	145 [Hao & Wu, 2012]
C4000.5	262 (**259***)	271 [Porumbel et al, 2010b]	259 [Hao & Wu, 2012]
C2000.9	**400**	413 [Wu & Hao, 2012]	408 [Hao & Wu, 2012]
Flat1000_76_0	**81**	82 [Lü & Hao, 2010]	81 [Hao & Wu, 2012]

*We found 259-colourings for C4000.5 by performing quantum annealing on a residual graph obtained from the independent set extraction experiments in [Wu & Hao, 2012] where only 260-colouring were found

Evolutionary-Tabu algorithms maintain a population of colourings that are iteratively improved by Tabu search and periodic crossover operations. The implementations in [Galinier & Hao, 1999; Lü & Hao, 2010; Porumbel et al, 2010b] provide some of the best results in the literature. They are discussed in more detail Section 3.5 in Chapter 3 and Section 4.4.6 in Chapter 4. Extraction pre-processing involves seeking large independent sets that can be removed from a graph to allow smaller residual graphs to be coloured with more conventional methods. The best results from this approach are found in [Wu and Hao, 2012; Hao and Wu, 2012].

Table 6.2: Detailed quantum annealing results

Graph	k	PT	Γ	Attempted	Accepted	Time	Success
DSJC500.5	**47**	0.30	0.70	2.8×10^{11}	4.1×10^{9}	36 min	2/10
DSJC1000.5	**82**	0.31	0.70	5.6×10^{11}	4.4×10^{9}	1.2 hr	10/10
DSJC1000.9	**222**	0.20	0.40	5.9×10^{11}	3.1×10^{9}	1.1 hr	6/10
C2000.5	146	0.32	0.65	2.5×10^{12}	1.1×10^{10}	5.4 hr	5/5
	145	0.32	0.69	1.4×10^{13}	6.7×10^{10}	31.6 hr	2/2
C4000.5	270	0.28	0.51	1.2×10^{14}	2.0×10^{11}	11 days	1/1
	262	0.28	0.57	1.3×10^{15}	1.8×10^{12}	4 mo.	1/1
C2000.9	403	0.18	0.29	1.2×10^{13}	2.0×10^{10}	24 hr	5/5
	402	0.18	0.295	2.8×10^{13}	4.5×10^{10}	54 hr	1/1
	401	0.18	0.30	8.8×10^{13}	1.4×10^{11}	174 hr	1/1
	400	0.18	0.31	1.3×10^{14}	2.0×10^{11}	505 hr	1/1
Flat1000_76_0	**81**	0.31	0.70	1.0×10^{12}	8.7×10^{9}	2.2 hr	10/10

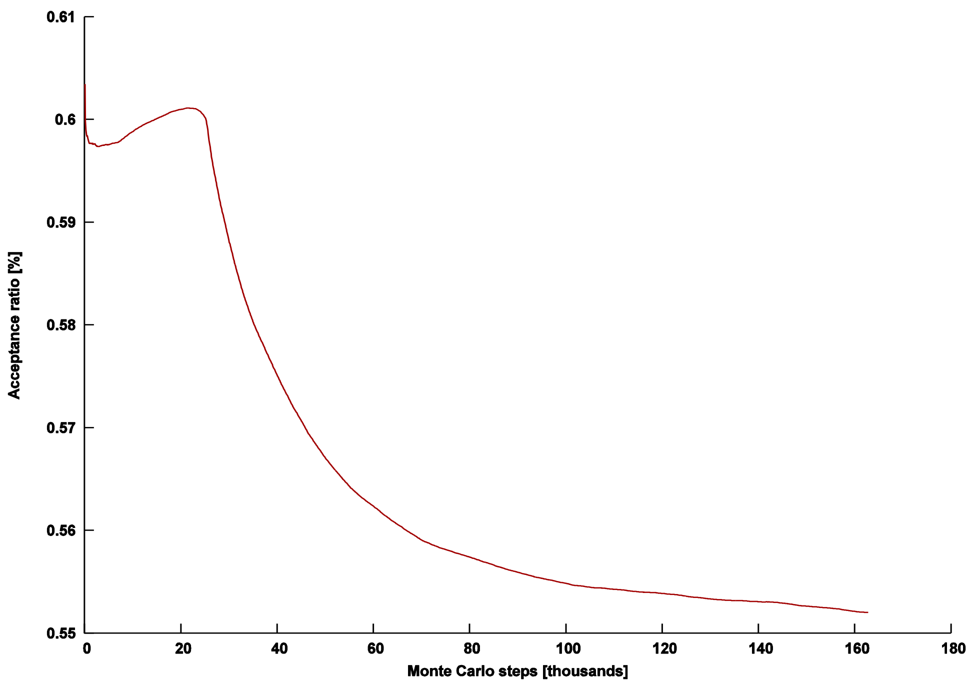

Figure 6.4: An ineffective simulation for (G = C2000.5, k = 146) with an increased temperature PT = 0.35. The acceptance ratio persistently declines in a manner very similar to classical annealing.

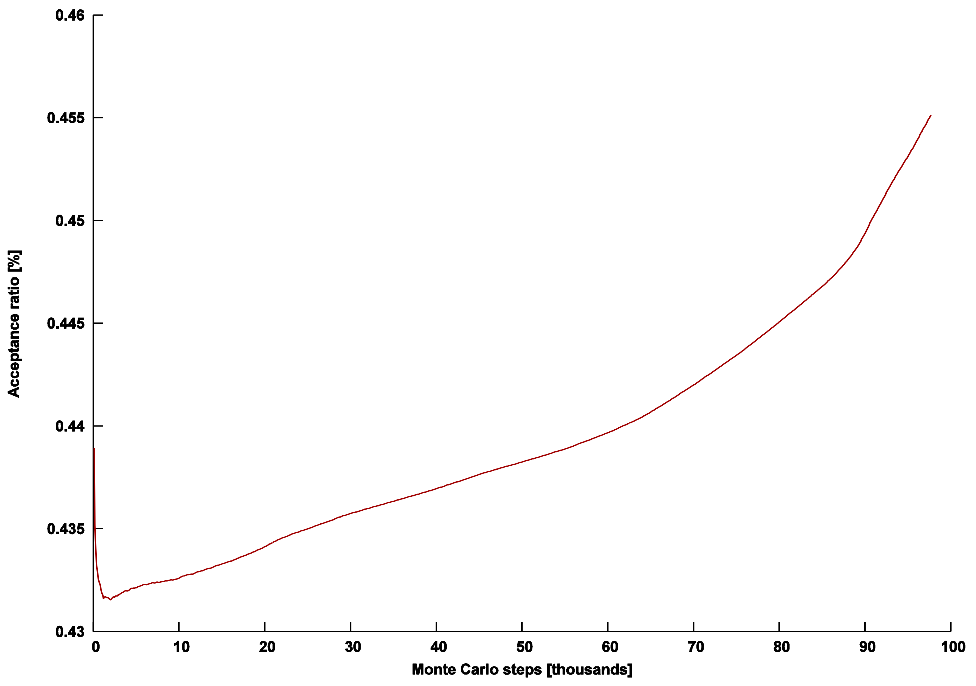

Figure 6.5: Acceptance ratio plot for (G = C2000.5, k = 146) with Γ = 0.65 and PT = 0.32. A successful simulation shows a continuously rising pattern for the acceptance ratio over time.

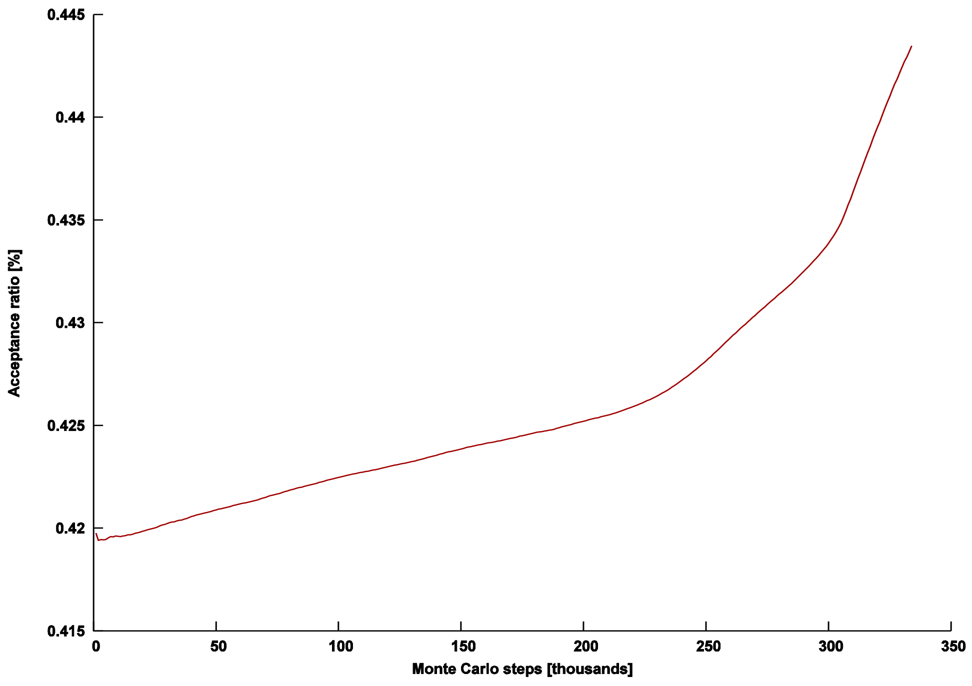

Figure 6.6: Acceptance ratio plot for (G = C2000.5, k = 145) with $\Gamma = 0.69$ and $PT = 0.32$**.** A successful simulation shows a continuously rising pattern for the acceptance ratio over time.

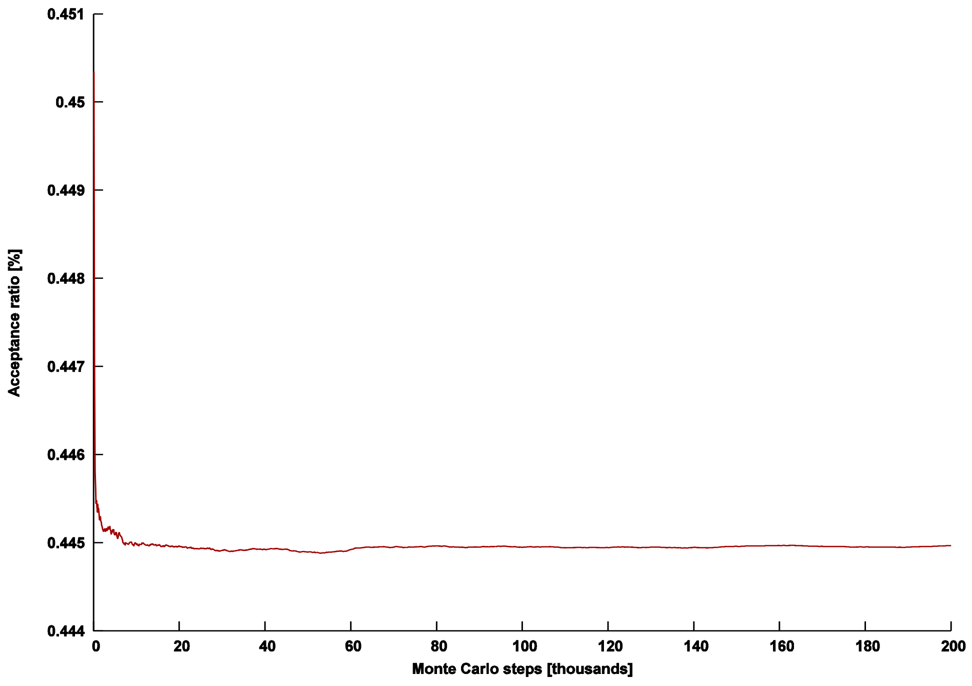

Figure 6.7: Acceptance ratio plot for (G = C2000.5, k = 146) with $\Gamma = 0.8$ and $PT = 0.32$. When the field strength is set higher than the critical value of $\Gamma = 0.65$, the simulation becomes ineffective. It also shows a persistent stagnation in the evolution of the acceptance ratio.

But rather like the situation of DSJC1000.5, the strategy stopped working when the instance with $k = 47$ was attempted, no matter the value of Γ. Success was only achieved with a continuously rising acceptance ratio after the temperature was reduced to $PT = 0.3$. The value of the field strength was kept the same at $\Gamma = 0.7$. We also carried out experiments using two larger $G(n,p)$ graphs with $p = 0.5$, namely C2000.5 and C4000.5. These two graphs with 2000 and 4000 vertices respectively, are rarely used in experiments because of the computational challenges created by their size. The graph C2000.5 has about a million edges, while C4000.5 has about four million edges. Until recently, the best result for C2000.5 of 150-colourings was reported in [Morgenstern, 1996]. Well over a decade after that result, 148-colourings were found using Evolutionary-Tabu algorithms [Lü & Hao, 2010; Porumbel et al., 2010b].

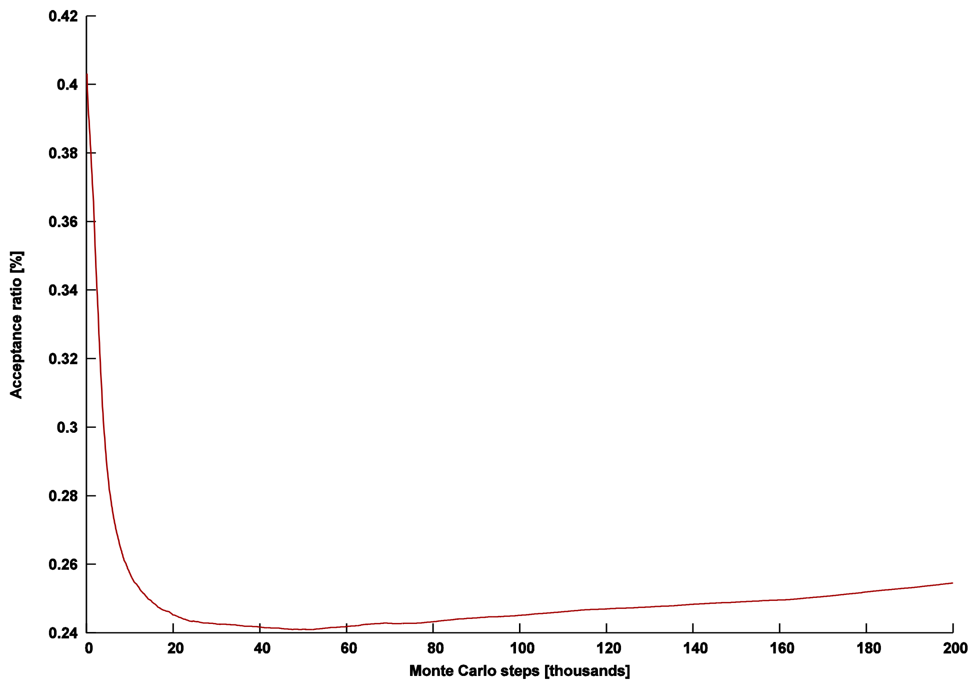

Figure 6.8: Acceptance ratio plot for (G = C2000.5, k = 146) with $\Gamma = 0.5$ and $PT = 0.32$. The simulation becomes ineffective when the field strength is set lower than the critical value of $\Gamma = 0.65$. The evolution of the acceptance ratio shows a long period of decline and a very weak growth afterwards.

Subsequently, the pre-processing technique of set extraction was revisited and used to obtain an improved result of 146-colourings [Wu & Hao, 2012] using the Evolutionary-Tabu algorithm in [Lü & Hao, 2010] as the main colouring algorithm. Afterwards, a revised algorithm consisting of iterating between the pre-processing procedure and the Evolutionary-Tabu algorithm was able to find 145-colourings. Any graph colouring algorithm can employ set extraction pre-processing. However, depending on the structure of the graph, this does not always lead to better results, and in fact one could obtain worse results by using the pre-processing. Medium density $G(n,p)$ graphs with $p \cong 0.5$ have been known to perform very well with set extraction pre-processing [Bollobás & Thomason, 1985; Chams et al., 1987; Hertz & Werra, 1987; Fleurent & Ferland, 1996; Morgenstern, 1996]. Large independent sets that are difficult to find by the main colouring procedure can be identified during the pre-processing stage. Additionally the random nature of the graph makes it less likely that the extraction of large independent sets before the colouring of a residual graph actually damages the prospects of achieving a good overall colouring. This is in contrast to geometric random graphs [Penrose, 2003], which exhibit more structure. It was observed in [Wu & Hao, 2012; Hao & Wu, 2012] that set extraction pre-processing was not helpful and was even counterproductive for geometric random graphs such as R1000.5 from the DIMACS benchmarks. The good behaviour of set extraction for $G(n,p)$ with $p = 0.5$ was further confirmed in [Wu & Hao, 2012] with the discovery of 260-colourings for C4000.5, when prior to that, the best available had been 272-colourings [Lü & Hao, 2010] and 271-colourings [Porumbel et al., 2010b] with Evolutionary-Tabu algorithms.

As can be seen from Table 6.1 and Table 6.2, quantum annealing with a random initial start can obtain 146-colourings as well as 145-colourings, with both instances working with a temperature of $PT = 0.32$. Like-for-like competition from Evolutionary-Tabu algorithms without set extraction pre-processing appears to be unable to match this, reaching their limit at 148-colourings [Lü & Hao, 2010; Porumbel et al., 2010b]. In terms of parameter tuning for quantum annealing on the C2000.5 graph, similar patterns to those of DSJC1000.5 are observed. Figure 6.4 shows the effect of setting the value of the temperature to $PT = 0.35$, which turns out to be too large. This results in a declining and stagnating pattern for the acceptance ratio. In contrast, the lower temperature $PT = 0.32$ results in the continuously increasing acceptance ratio depicted in Figure 6.5. While simulations with $PT = 0.32$ can repeatedly solve $(G = \text{C2000.5},\ k = 146)$ in about 100,000 Monte Carlo steps, we verified that simulations with $PT = 0.35$ repeatedly fail to find a solution even when allowed up to 500,000 Monte Carlo steps. Similarly, for the more difficult instance $(G = \text{C2000.5}, k = 145)$, the lower temperature results in success and a continuously increasing acceptance ratio shown in Figure 6.6. So far, the temperature has been varied while all other conditions have been kept constant, such as keeping the value of the field strength at $\Gamma = 0.65$ for $k = 146$. To show that this is important, two additional sets of experiments were run with higher and a lower values of $\Gamma = 0.8$ and $\Gamma = 0.5$ respectively. The acceptance ratio patterns produced are depicted in Figure 6.7 and Figure 6.8. They are significantly different from the successful ones in Figure 6.5 and Figure 6.6, and no solutions are found.

Quantum annealing can also outperform the best representatives of Evolutionary-Tabu on C4000.5, improving over the 271-colourings in [Porumbel et al., 2010b] by finding 270-colourings and even 262-colourings in the long run. However we found it necessary to follow suit with extraction pre-processing in order to match the best results of 260-colourings [Wu & Hao, 2012] and 259-colourings [Hao & Wu, 2012]. Our quantum annealing results for C4000.5 are presented in Table 6.1 and Table 6.2.

The graph C2000.9 is a $G(2000{,}0.9)$ graph from the DIMACS benchmark for which 409-colourings and 408-colourings were reported in [Wu & Hao, 2012] and [Hao & Wu, 2012] respectively, using set extraction pre-processing. When the standard random initialization was used, a worse result of 413 was obtained by their Evolutionary-Tabu algorithm [Wu & Hao, 2012]. Remarkably quantum annealing is able to find 400-colourings for C2000.9, beating the previously best known result in [Wu & Hao, 2012] by eight colours. The comparatively weaker effect of the set extraction pre-processing on this much denser instance is at least partially explained by the fact that there is very little variance between the sizes of maximal independent sets found in such a dense random graph. For example, the two known largest independent set sizes that our algorithm and any other algorithm are currently able to find are 6 and 5. These independent sets of C2000.9 are very numerous and very easy to find in comparison to those of size 18 in C4000.5 for example. The problem encountered when colouring C4000.5 is mainly that of finding a large number of mutually exclusive independent sets of size 18 and 17. Whichever of those independent sets gets extracted is not usually of importance to the quality of the final result. However in the case of C2000.9, the main problem is that of finding the right combination of mutually independent sets of size 6 and 5 to extract, such that the colouring of the resulting residual graph still leads to a high quality colouring. As a result, the set extraction pre-processing is of less help, and the burden falls unto the main colouring algorithm to find the right combinations of mutually exclusive independent sets. It is here that the quantum annealing algorithm appears to do a better job than MACOL [Lü & Hao, 2010], the Evolutionary-Tabu colouring algorithm used in [Wu & Hao, 2012; Hao & Wu, 2012].

The flat graph Flat1000_76_0, even though not an Erdös-Rényi random graph has been known to behave similarly to DSJC1000.5 in terms of parameter settings, computation effort and chromatic number upper bounds when coloured by metaheuristic algorithms [Galinier & Hao, 1999]. The graph Flat1000_76_0 also consists of the same number of vertices and is of similar density to DSJC1000.5. In chapter 4 and chapter 5, we used the same parameter settings $PT = 0.36$ and $\Gamma = 0.67$ to find 83-colourings and 82-colourings of

DSJC1000.5 and Flat1000_76_0 respectively. It was observed during experimentation that Flat1000_76_0 exhibits the same problems when an attempt is made to find an 81-colouring. Keeping $PT = 0.36$ and setting an increased value for Γ did not work. After the observation that setting a reduced $PT = 0.31$ improves the results of DSJC1000.5, it was natural to test the same idea on Flat1000_76_0 to check if it would work here as well. Experiments showed that it indeed does, with a similar 100% success rate out of 10 runs, and with a similar computational effort. This result is listed in Table 6.1 and Table 6.2.

6.6 Conclusion

The aim of this chapter was to investigate tuning patterns peculiar to random graphs in order to provide an understanding of how quantum annealing can be made to tackle graph colouring more effectively. The evolution of the acceptance ratio exhibited during simulations was found to be related to whether difficult instances were solved or not. As a direct result of this finding, some problem instances that had resisted solution for about two decades were solved.

Chapter 7

Conclusions and Perspectives

7.1 Conclusions

We have developed a Monte Carlo quantum annealing algorithm for the graph colouring problem. In the introductory chapter 1, we set out to present:

- The first successful quantum annealing algorithm for the graph *k*-colouring problem by using an effecting Boolean representation, and finding fast incremental calculations for the resulting more complex cost function for quantum annealing
- Enhancements of the quantum annealing algorithm by the incorporation of tuning simplification, replica spacing and parallelization
- Insights into the optimal tuning for random graphs following from observations on the evolution of the acceptance ratio

The following paragraphs revisit these three statements in turn.

In chapter 4, an alternative problem representation was introduced to counteract difficulties that often arise due to symmetries in the search space induced by graph colouring problems. The unconventional problem representation though very useful, presented computational challenges for calculating energy changes due to an involved kinetic energy expression. Fast incremental techniques were found, which made the presented quantum annealing algorithm competitive. This included the provision of an easily computable upper bound to the kinetic energy that could be used in place of the actual value most of the time without introducing any errors. The algorithm became the first in the literature to colour the benchmark graph DSJC1000.9 with 222 colours. Quantum annealing was competitive with the best algorithms on the overall DIMACS benchmarks on the most relevant criterion of the best chromatic number upper bound found on each graph. This suggests that quantum annealing is a viable approach to population based metaheuristics as an alternative to current Evolutionary-hybrid approaches.

In chapter 5, several enhancements and variations to the basic quantum annealing algorithm for graph colouring were presented. Inspired by analogous schedules for some classical annealing algorithms, the field strength parameter was fixed rather than decremented, and this was demonstrated to simplify tuning and increase robustness. The fixed parameters ensured that the opportunity to find a new upper bound of 97 for the chromatic number of a well known graph named Latin_square_10 was not missed. A replica spacing scheme similar to those used in some evolutionary algorithms was also introduced for quantum annealing for the first time. This demonstrably resulted in an algorithm more resilient to premature convergence, thereby allowing solutions to larger and more difficult problem instances. As a result, quantum annealing was able to improve on recent results of some Evolutionary algorithms [Lü & Hao, 2010; Porumbel et al., 2010b] by finding 147-colourings for the graph C2000.5. A parallelized version of the algorithm was also presented, thereby enabling the reporting of faster computing times than would usually be possible.

In chapter 6, tuning patterns were investigated for the colouring of random graphs. It was found that the most difficult instances could be solved by setting parameters that result in a continuously increasing acceptance ratio for quantum annealing. As a result, several new chromatic number upper bounds were discovered, including 47 for DSJC500.5 and 82 for DSJC1000.5. These graphs had not had their upper bounds improved in almost two decades of study in the literature. We also obtained a 400-colouring for C2000.9, beating the nearest competitor [Hao & Wu, 2012] by eight colours. This demonstrated that despite the merits of the set extraction approaches advanced in [Wu & Hao, 2012] and [Hao & Wu, 2012], it can still be outperformed in several cases by the strong population interaction provided by the quantum annealing approach.

7.2 Further Work

A good lead for future work is the application of quantum annealing to other combinatorial optimization problems using techniques developed here. The techniques that might be transferable include the casting of problem representations to a set of constrained Boolean variables similar to ours, the use of the newly introduced errorless surrogate function swindle, and the structure of our parallelized algorithm. In the cases where the new problem is closely related to graph colouring, such as list colouring [Diestel, 2000] and exam timetabling [Sabar et al., 2012], the Boolean representation presented in chapter 1 could be directly applicable with small modifications. For entirely different combinatorial optimization problems such as the quadratic assignment problem [James et al., 2009], appropriate problem specific representations and procedures may need to be developed, and its own favourable tuning patterns may need to be found by using similar investigative approaches to the one presented in chapter 6. Our parallel algorithm structure presented in chapter 5 is likely to be applicable to different problems. The future of quantum annealing as a metaheuristic appears promising.

References

[Titiloye & Crispin, 2011a]
Titiloye, O., & Crispin, A. (2011). Quantum annealing of the graph coloring problem. *Discrete Optimization*, *8*(2), 376-384.

[Titiloye & Crispin, 2011b]
Titiloye, O., & Crispin, A. (2011). Graph coloring with a distributed hybrid quantum annealing algorithm. *Agent and Multi-Agent Systems: Technologies and Applications*, 553-562.

[Titiloye & Crispin, 2012]
Titiloye, O., & Crispin, A. (2012). Parameter tuning patterns for random graph coloring with quantum annealing. *PLOS ONE*, *7*(11), e50060.

[Aarts et al., 1997]
Aarts, E. H., Korst, J. H., & Van Laarhoven, P. J. (1997). Simulated annealing. *Local search in combinatorial optimization*, 91-120.

[Achlioptas & Coja-Oghlan, 2008]
Achlioptas, D., & Coja-Oghlan, A. (2008, October). Algorithmic barriers from phase transitions. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on* (pp. 793-802). IEEE.

[Achlioptas et al., 2005]
Achlioptas, D., Naor, A., & Peres, Y. (2005). Rigorous location of phase transitions in hard optimization problems. *Nature*, *435*(7043), 759-764.

[Appel & Haken, 1977]
Appel, K., & Haken, W. (1977). Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics*, *21*(3), 429-490.

[Bäck, 1996]
Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA.

[Barahona, 1982]
Barahona, F. (1982). On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General*, *15*(10), 3241.

[Battaglia et al., 2005]
Battaglia, D. A., Santoro, G. E., & Tosatti, E. (2005). Optimization by quantum annealing: lessons from hard satisfiability problems. *Physical Review E*, *71*(6), 066707.

[Blöchliger & Zufferey, 2008]
Blöchliger, I., & Zufferey, N. (2008). A graph coloring heuristic using partial solutions and a reactive tabu scheme. Computers & Operations Research, 35(3), 960-975.

[Bollobás & Thomason, 1985]
Bollobás, B., & Thomason, A. (1985, December). Random graphs of small order. In *Random graphs* (Vol. 83, pp. 47-97).

[Bollobás, 2001]
Bollobás, B. (2001). *Random graphs* (Vol. 73). Cambridge university press.

[Boman et al., 2005]
Boman, E., Bozdag, D., Catalyurek, U., Gebremedhin, A., & Manne, F. (2005). A scalable parallel graph coloring algorithm for distributed memory computers. Euro-Par 2005 Parallel Processing, 613-613.

[Brélaz, 1979]
Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, *22*(4), 251-256.

[Briggs et al., 1989]
Briggs, P., Cooper, K. D., Kennedy, K., & Torczon, L. (1989, June). Coloring heuristics for register allocation. In *ACM SIGPLAN Notices* (Vol. 24, No. 7, pp. 275-284). ACM.

[Brown, 1972]
Brown, J. R. (1972). Chromatic scheduling and the chromatic number problem. *Management Science*, *19*(4-Part-1), 456-463.

[Černý, 1985]
Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, *45*(1), 41-51.

[Chaitin, 1982]
Chaitin, G. J. (1982, June). Register allocation & spilling via graph coloring. In *ACM Sigplan Notices* (Vol. 17, No. 6, pp. 98-105). ACM.

[Chams et al., 1987]
Chams, M., Hertz, A., & Werra, D. D. (1987). Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research*, *32*(2), 260-266.

[Chow & Hennessy, 1984]
Chow, F., & Hennessy, J. (1984, June). Register allocation by priority-based coloring. In *ACM Sigplan Notices* (Vol. 19, No. 6, pp. 222-232). ACM.

[Cipra, 1987]
Cipra, B. A. (1987). An introduction to the Ising model. *American Mathematical Monthly*, *94*(10), 937-959.

[Cobham, 1965]
Cobham, A. (1965). The intrinsic computational difficulty of functions. In *Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science* (pp. 24-30).

[Cohn & Fielding, 1999]

Cohn, H., & Fielding, M. (1999). Simulated annealing: searching for an optimal temperature schedule. *SIAM Journal on Optimization*, *9*(3), 779-802.

[Connolly, 1990]
Connolly, D. T. (1990). An improved annealing scheme for the QAP. *European Journal of Operational Research*, *46*(1), 93-100.

[Cook, 1971]
Cook, S. A. (1971, May). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing* (pp. 151-158). ACM.

[Crainic & Toulouse, 2003]
Crainic, T., & Toulouse, M. (2003). Parallel strategies for meta-heuristics. Handbook of metaheuristics, 475-513.

[Crainic et al., 1997]
Crainic, T. G., Toulouse, M., & Gendreau, M. (1997). Toward a taxonomy of parallel tabu search heuristics. INFORMS Journal on Computing, 9(1), 61-72.

[Croes, 1958]
Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, *6*(6), 791-812.

[Culberson & Gent, 2001]
Culberson, J., & Gent, I. (2001). Frozen development in graph coloring. *Theoretical Computer Science*, *265*(1), 227-264.

[Culberson & Luo, 1996]
Culberson, J. C., & Luo, F. (1996). Exploring the k-colorable landscape with iterated greedy. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge*, *26*, 245-284.

[Culberson et al., 1995]
Culberson, J., Beacham, A., & Papp, D. (1995, September). Hiding our colors. In *CP'95 Workshop on Studying and Solving Really Hard Problems* (pp. 31-42).

[Cung et al., 2002]
Cung, V. D., Martins, S. L., Ribeiro, C. C., & Roucairol, C. (2002). Strategies for the parallel implementation of metaheuristics. Essays and surveys in metaheuristics, 15.

[Das & Chakrabarti, 2008]
Das, A., & Chakrabarti, B. K. (2008). Colloquium: Quantum annealing and analog quantum computation. *Reviews of Modern Physics*, *80*(3), 1061.

[Diestel, 2000]
Diestel, R. (2000). Graph Theory, volume 173 of. *Graduate texts in mathematics*, 24-26.

[Dorne & Hao, 1995]
Dorne, R., & Hao, J. K. (1995, November). An evolutionary approach for frequency assignment in cellular radio networks. In *Evolutionary Computation, 1995., IEEE International Conference on* (Vol. 2, pp. 539-544). IEEE.

[Dorne & Hao, 1998a]
Dorne, R., & Hao, J. K. (1998). Tabu search for graph coloring, T-colorings and set T-colorings. *Meta-heuristics: Advances and trends in local search paradigms for optimization*, 77-92.

[Dorne & Hao, 1998b]
Dorne, R., & Hao, J. K. (1998). A new genetic local search algorithm for graph coloring. In Parallel Problem Solving from Nature—PPSN V (pp. 745-754). Springer Berlin/Heidelberg.

[Edwards & Anderson, 1975]
Edwards, S. F., & Anderson, P. W. (1975). Theory of spin glasses. *Journal of Physics F: Metal Physics*, *5*(5), 965.

[Eglese, 1990]
Eglese, R. W. (1990). Simulated annealing: a tool for operational research. European journal of operational research, 46(3), 271-281.

[Eiben & Bäck, 1997]
Eiben, A. E., & Bäck, T. (1997). Empirical investigation of multiparent recombination operators in evolution strategies. *Evolutionary Computation*, *5*(3), 347-365.

[Erdös & Rényi, 1959]
Erdös, P., & Rényi, A. (1959). On random graphs I. *Publ. Math. Debrecen*, *6*, 290-297.

[Erdös & Rényi, 1960]
Erdős, P., & Rényi, A. (1960). On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, *5*, 17-61.

[Falkenauer, 1994]
Falkenauer, E. (1994). A new representation and operators for genetic algorithms applied to grouping problems. *Evolutionary Computation*, *2*(2), 123-144.

[Falkenauer, 1998]
Falkenauer, E. (1998). *Genetic algorithms and grouping problems*. John Wiley & Sons, Inc..

[Farhi et al., 2001]
Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., & Preda, D. (2001). A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, *292*(5516), 472-475.

[Felten et al., 1985]
Felten, E., Karlin, S., & Otto, S. W. (1985, August). The traveling salesman problem on a hypercubic, mimd computer. In Proc. 1985 Parallel Processing Conf (pp. 6-10).

[Fielding, 2000]
Fielding, M. (2000). Simulated annealing with an optimal fixed temperature. *SIAM Journal on Optimization*, *11*(2), 289-307.

[Finnila et al., 1994]

Finnila, A. B., Gomez, M. A., Sebenik, C., Stenson, C., & Doll, J. D. (1994). Quantum annealing: a new method for minimizing multidimensional functions. *Chemical physics letters*, *219*(5), 343-348.

[Fleurent & Ferland, 1996]
Fleurent, C., & Ferland, J. A. (1996). Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, *63*(3), 437-461.

[Flood, 1956]
Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, *4*(1), 61-75.

[Funabiki & Higashino, 2000]
Funabiki, N., & Higashino, T. (2000). A minimal-state processing search algorithm for graph coloring problems. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, *83*(7), 1420-1430.

[Galinier & Hao, 1999]
Galinier, P., & Hao, J. K. (1999). Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization*, *3*(4), 379-397.

[Galinier & Hertz, 2006]
Galinier, P., & Hertz, A. (2006). A survey of local search methods for graph coloring. *Computers & Operations Research*, *33*(9), 2547-2562.

[Galinier et al., 2008]
Galinier, P., Hertz, A., & Zufferey, N. (2008). An adaptive memory algorithm for the k-coloring problem. Discrete Applied Mathematics, 156(2), 267-279.

[Garey & Johnson, 1979]
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability, a guide to the theory of NP-Completeness* (Vol. 174). San Francisco, CA: Freeman.

[Garey et al., 1976]
Garey, M., Johnson, D., & So, H. (1976). An application of graph coloring to printed circuit testing. *Circuits and Systems, IEEE Transactions on*, *23*(10), 591-599.

[Gebremedhin & Manne, 2000]
Gebremedhin, A. H., & Manne, F. (2000). Scalable parallel graph coloring algorithms. *Concurrency - Practice and Experience*, *12*(12), 1131-1146.

[Gebremedhin et al., 2005]
Gebremedhin, A. H., Manne, F., & Pothen, A. (2005). What color is your Jacobian? Graph coloring for computing derivatives. *SIAM review*, *47*(4), 629-705.

[Geman & Geman, 1984]
Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6), 721-741.

[Gilbert, 1959]

Gilbert, E. N. (1959). Random graphs. *The Annals of Mathematical Statistics*, 1141-1144.

[Glover, 1989]
Glover, F. (1989). Tabu search—part I. *ORSA Journal on computing*, *1*(3), 190-206.

[Glover, 1990]
Glover, F. (1990). Tabu search—part II. *ORSA Journal on computing*, *2*(1), 4-32.

[Greene & Supowit, 1986]
Greene, J. W., & Supowit, K. J. (1986). Simulated annealing without rejected moves. IEEE transactions on computer-aided design of integrated circuits and systems, 5(1), 221-228.

[Gualandi et al., 2012]
Gualandi, S., & Malucelli, F. (2012). Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing*, *24*(1), 81-100.

[Gusfield, 2002]
Gusfield, D. (2002). Partition-distance: A problem and class of perfect graphs arising in clustering. *Information Processing Letters*, *82*(3), 159-164.

[Hajek, 1985]
Hajek, B. (1985, December). A tutorial survey of theory and applications of simulated annealing. In *Decision and Control, 1985 24th IEEE Conference on* (Vol. 24, pp. 755-760). IEEE.

[Hale, 1980]
Hale, W. K. (1980). Frequency assignment: Theory and applications. *Proceedings of the IEEE*, *68*(12), 1497-1514.

[Halldórsson, 1993]
Halldórsson, M. M. (1993). A still better performance guarantee for approximate graph coloring. *Information Processing Letters*, *45*(1), 19-23.

[Hamiez & Hao, 2004]
Hamiez, J. P., & Hao, J. K. (2004). 15 An analysis of solution properties of the graph coloring problem. Metaheuristics: Computer Decision-Making, 325.

[Hao & Wu, 2012]
Hao, J. K., & Wu, Q. (2012). Improving the extraction and expansion method for large graph coloring. Discrete Applied Mathematics.

[Hertz & Werra, 1987]
Hertz, A., & Werra, D. D. (1987). Using tabu search techniques for graph coloring. *Computing*, *39*(4), 345-351.

[Hillis & Steele, 1986]
Hillis, W. D., & Steele Jr, G. L. (1986). Data parallel algorithms. Communications of the ACM, 29(12), 1170-1183.

[Hoos & Stiitzle, 2000]
Hoos, H., & Stiitzle, T. (2000). SATLlB: An Online Resource for Research on SAT. *Sat*, 283.

[Hoos & Stiitzle, 2004]
Hoos, H. H., & Stützle, T. (2004). Stochastic local search: Foundations & applications. Morgan Kaufmann.

[Hossain & Steihaug, 2008]
Hossain, S., & Steihaug, T. (2008). Graph coloring in the estimation of sparse derivative matrices: Instances and applications. *Discrete Applied Mathematics*, *156*(2), 280-288.

[Ising, 1925]
Ising, E. (1925). A Contribution to the Theory of Ferromagnetism. *Z. Phys*, *31*(1), 253-258.

[James et al., 2009]
James, T., Rego, C., & Glover, F. (2009). A cooperative parallel tabu search algorithm for the quadratic assignment problem. European Journal of Operational Research, 195(3), 810-826.

[Johnson & McGeoch, 1997]
Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 215-310.

[Johnson and Trick, 1996]
Johnson, D. S., & Trick, M. A. (1996). *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993* (Vol. 26). Amer Mathematical Society.

[Johnson et al., 1991]
Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1991). Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Operations research*, *39*(3), 378-406.

[Johnson et al., 2011]
Johnson, M. W., Amin, M. H. S., Gildert, S., Lanting, T., Hamze, F., Dickson, N., ... & Rose, G. (2011). Quantum annealing with manufactured spins. *Nature*, *473*(7346), 194-198
.
[Johri & Matula, 1982]
Johri, A., & Matula, D. W. (1982). *Probabilistic bounds and heuristic algorithms for coloring large random graphs* (Master's thesis, S. M. U.).

[Jones & Plassmann, 1993]
Jones, M. T., & Plassmann, P. E. (1993). A parallel graph coloring heuristic. SIAM Journal on Scientific Computing, 14(3), 654-669.

[Jonker & Volgenant, 1986]
Jonker, R., & Volgenant, T. (1986). Improving the Hungarian assignment algorithm. *Operations Research Letters*, *5*(4), 171-175.

[Kadowaki & Nishimori, 1998]
Kadowaki, T., & Nishimori, H. (1998). Quantum annealing in the transverse Ising model. *Physical Review E*, *58*(5), 5355.

[Karp, 1972]
Karp, R.M. (1972). Reducibility among combinatorial optimization problems. In *Complexity of Computer Computations* (pp. 85-104). Plenum Press New York.

[Kirkpatrick et al., 1983]
Kirkpatrick, S., Gelatt, C.D., & Vecchi, M. P. (1983). Optimization by simmulated annealing. *science*, *220*(4598), 671-680.

[Kokosiński et al., 2004]
Kokosiński, Z., Kołodziej, M., & Kwarciany, K. (2004). Parallel genetic algorithm for graph coloring problem. *Computational Science-ICCS 2004*, 215-222.

[Kuhn, 1955]
Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, *2*(1-2), 83-97.

[Kurihara et al., 2009]
Kurihara, K., Tanaka, S., & Miyashita, S. (2009, June). Quantum annealing for clustering. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 321-328). AUAI Press.

[Lam & Delosme, 1988]
Lam, J., & Delosme, J. M. (1988, June). Performance of a new annealing schedule. In *Proceedings of the 25th ACM/IEEE Design Automation Conference* (pp. 306-311). IEEE Computer Society Press.

[Lecina, 2011]
Lecina Casas, D. (2011). Quantum annealing of a hard combinatorial problem. http://hdl.handle.net/2099.1/11313

[Lecuyer & Simard, 2007]
Lecuyer, P., & Simard, R. (2007). TestU01: AC library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, *33*(4).

[Lee & Berne, 2000]
Lee, Y. H., & Berne, B. J. (2000). Global optimization: quantum thermal annealing with path integral Monte Carlo. *The Journal of Physical Chemistry A*, *104*(1), 86-95.

[Leighton, 1979]
Leighton, F. T. (1979). A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, *84*(6), 489-506.

[Lewandowski & Condon, 1996]
Lewandowski, G., & Condon, A. (1996). Experiments with parallel graph coloring heuristics and applications of graph coloring. *Johnson & Trick*, 309.

[Lü & Hao, 2010]
Lü, Z., & Hao, J. K. (2010). A memetic algorithm for graph coloring. *European Journal of Operational Research*, *203*(1), 241-250.

[Malaguti et al., 2008]
Malaguti, E., Monaci, M., & Toth, P. (2008). A metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing*, *20*(2), 302-316.

[Malaguti et al., 2011]
Malaguti, E., Monaci, M., & Toth, P. (2011). An exact approach for the vertex coloring problem. *Discrete Optimization*, *8*(2), 174-190.

[Marinari et al., 1998]
Marinari, E., Parisi, G., & Ruiz-Lorenzo, J. J. (1998). Phase structure of the three-dimensional Edwards-Anderson spin glass. *Physical Review B*, *58*(22), 14852.

[Marques-Silva, 2008]
Marques-Silva, J. (2008, May). Practical applications of Boolean satisfiability. In *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on* (pp. 74-80). IEEE.

[Marsaglia & Tsang, 2002]
Marsaglia, G., & Tsang, W. W. (2002). Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, *7*(3), 1-9.

[Marsaglia, 2003]
Marsaglia, G. (2003). Xorshift rngs. *Journal of Statistical Software*, *8*(14), 1-6.

[Martoňák et al., 2002]
Martoňák, R., Santoro, G. E., & Tosatti, E. (2002). Quantum annealing by the path-integral Monte Carlo method: The two-dimensional random Ising model. *Physical Review B*, *66*(9), 094203.

[Martoňák et al., 2004]
Martoňák, R., Santoro, G. E., & Tosatti, E. (2004). Quantum annealing of the traveling-salesman problem. *Physical Review E*, *70*(5), 057701.

[Matsumoto & Nishimura, 1998]
Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *8*(1), 3-30.

[Maucher et al., 2008]
Maucher, M., Schöning, U., & Kestler, H. A. (2008). *An empirical assessment of local and population based search methods with different degrees of pseudorandomness*. University of Ulm. Faculty of engineering and computer science.

[Mehta, 1981]
Mehta, N. K. (1981). The application of a graph coloring method to an examination scheduling problem. *Interfaces*, *11*(5), 57-65.

[Méndez-Díaz & Zabala, 2008]
Méndez-Díaz, I., & Zabala, P. (2008). A cutting plane algorithm for graph coloring. Discrete Applied Mathematics, 156(2), 159-179.

[Metropolis et al., 1953]
Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, *21*, 1087.

[Mézard et al., 2002]
Mézard, M., Parisi, G., & Zecchina, R. (2002). Analytic and algorithmic solution of random satisfiability problems. *Science*, *297*(5582), 812-815.

[Mitra et al., 1986]
Mitra, D., Romeo, F., & Sangiovanni-Vincentelli, A. (1986). Convergence and finite-time behavior of simulated annealing. *Advances in Applied Probability*, 747-771.

[Monasson et al., 1999]
Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., & Troyansky, L. (1999). Determining computational complexity from characteristic 'phase transitions'. *Nature*, *400*(6740), 133-137.

[Morgenstern, 1996]
Morgenstern, C. (1996). Distributed coloration neighborhood search. *Discrete Mathematics and Theoretical Computer Science*, *26*, 335-358.

[Moscato, 1989]
Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, *826*, 1989.

[Moscato, 2003]
Moscato, P., & Cotta, C. (2003). A gentle introduction to memetic algorithms. *Handbook of metaheuristics*, 105-144.

[Moscato, 2005]
Moscato, P., Berretta, R., & Cotta, C. (2005). Memetic algorithms. *Wiley Encyclopedia of Operations Research and Management Science*.

[Pál, 1996]
Pál, K. F. (1996). The ground state energy of the Edwards-Anderson Ising spin glass with a hybrid genetic algorithm. *Physica A: Statistical Mechanics and its Applications*, *223*(3), 283-292.

[Panneton & L’ecuyer, 2005]
Panneton, F., & L'ecuyer, P. (2005). On the xorshift random number generators. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *15*(4), 346-361.

[Penrose, 2003]
Penrose, M. (2003). Random geometric graphs (Vol. 5). Oxford, UK:: Oxford University Press.

[Porumbel et al., 2008]

Porumbel, D., Hao, J. K., & Kuntz, P. (2008). A study of evaluation functions for the graph K-coloring problem. In *Artificial Evolution* (pp. 124-135). Springer Berlin/Heidelberg.

[Porumbel et al., 2010a]
Porumbel, D. C., Hao, J. K., & Kuntz, P. (2010). A search space "cartography" for guiding graph coloring heuristics. *Computers & Operations Research*, *37*(4), 769-778.

[Porumbel et al., 2010b]
Porumbel, D. C., Hao, J. K., & Kuntz, P. (2010). An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. *Computers & Operations Research*, *37*(10), 1822-1832.

[Porumbel et al., 2011]
Porumbel, D., Hao, J. K., & Kuntz, P. (2011, July). Spacing memetic algorithms. In Proceedings of Gecco (pp. 1061-1068).

[Ramani et al., 2004]
Ramani, A., Aloul, F. A., Markov, I. L., & Sakallah, K. A. (2004, February). Breaking instance-independent symmetries in exact graph coloring. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings* (Vol. 1, pp. 324-329). IEEE.

[Reinelt, 1991]
Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. *ORSA journal on computing*, *3*(4), 376-384.

[Rochat & Taillard, 1995]
Rochat, Y., & Taillard, É. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. Journal of heuristics, 1(1), 147-167.

[Sabar et al., 2012]
Sabar, N. R., Ayob, M., Qu, R., & Kendall, G. (2012). A graph coloring constructive hyper-heuristic for examination timetabling problems. *Applied Intelligence*, *37*(1), 1-11.

[San Segundo, 2012]
San Segundo, P. (2012). A new DSATUR-based algorithm for exact vertex coloring. *Computers and Operations Research*, *39*(7), 1724.

[Santoro et al., 2002]
Santoro, G. E., Martoňák, R., Tosatti, E., & Car, R. (2002). Theory of quantum annealing of an Ising spin glass. *Science*, *295*(5564), 2427-2430.

[Sato et al., 2009]
Sato, I., Kurihara, K., Tanaka, S., Nakagawa, H., & Miyashita, S. (2009, June). Quantum annealing for variational Bayes inference. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 479-486). AUAI Press.

[Schraudolph, 1999]
Schraudolph, N. N. (1999). A fast, compact approximation of the exponential function. *Neural Computation*, *11*(4), 853-862.

[Selman et al., 1993]
Selman, B., Kautz, H., & Cohen, B. (1993). Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*, *26*, 521-532.

[Stanley, 2011]
Stanley, R. P. (2011). *Enumerative combinatorics* (Vol. 49). Cambridge university press.

[Stella et al., 2005]
Stella, L., Santoro, G. E., & Tosatti, E. (2005). Optimization by quantum annealing: Lessons from simple cases. *Physical Review B*, *72*(1), 014303.

[Strenski & Kirkpatrick, 1991]
Strenski, P. N., & Kirkpatrick, S. (1991). Analysis of finite length annealing schedules. *Algorithmica*, *6*(1), 346-366.

[Subhlok et al., 1993]
Subhlok, J., Stichnoth, J. M., O'hallaron, D. R., & Gross, T. (1993). Exploiting task and data parallelism on a multicomputer. ACM SIGPLAN Notices, 28(7), 13-22.

[Suzuki, 1976]
Suzuki, J. (1976). Relationship between d-dimensional quantal spin systems and (d+ 1)-dimensional Ising systems. *Prog. Theor. Phys*, *56*(5), 1454-1469.

[Thompson, 1979]
Thompson, C. J. (1979). *Mathematical statistical mechanics*. Princeton University Press.

[Tovey, 1988]
Tovey, C. (1988). Simulated simulated annealing. *American Journal of Mathematical and Management Sciences*, *8*(3), 389-407.

[Verhoeven & Aarts, 1995]
Verhoeven, M. G., & Aarts, E. H. (1995). Parallel local search. Journal of Heuristics, 1(1), 43-65.

[Wright, 1990]
Wright, M. B. (1990). Speeding up the Hungarian algorithm. *Computers & Operations Research*, *17*(1), 95-96.

[Wu & Hao, 2012]
Wu, Q., & Hao, J. K. (2012). Coloring large graphs based on independent set extraction. *Computers & Operations Research*, *39*(2), 283-290.

[Xie & Liu, 2009]
Xie, X. F., & Liu, J. (2009). Graph coloring by multiagent fusion search. Journal of combinatorial optimization, 18(2), 99-123.

[Zdeborová & Krząkała, 2007]
Zdeborová, L., & Krząkała, F. (2007). Phase transitions in the coloring of random graphs. *Physical Review E*, *76*(3), 031131.

[Zuckerman, 2006]
Zuckerman, D. (2006, May). Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing* (pp. 681-690). ACM.