# QUANTUM ANNEALING FOR VEHICLE ROUTING AND SCHEDULING PROBLEMS

A SYRICHAS

PhD 2019

# Quantum Annealing for Vehicle Routing and Scheduling Problems

ALEX SYRICHAS

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

Manchester Metropolitan University
Department of Computing and Mathematics

2019

## Abstract

Metaheuristic approaches to solving combinatorial optimization problems have many attractions. They sidestep the issue of combinatorial explosion; they return good results; they are often conceptually simple and straight forward to implement. There are also shortcomings. Optimal solutions are not guaranteed; choosing the metaheuristic which best fits a problem is a matter of experimentation; and conceptual differences between metaheuristics make absolute comparisons of performance difficult. There is also the difficulty of configuration of the algorithm - the process of identifying precise values for the parameters which control the optimization process.

Quantum annealing is a metaheuristic which is the quantum counterpart of the well known classical Simulated Annealing algorithm for combinatorial optimization problems. This research investigates the application of quantum annealing to the Vehicle Routing Problem, a difficult problem of practical significance within industries such as logistics and workforce scheduling. The work devises spin encoding schemes for routing and scheduling problem domains, enabling an effective quantum annealing algorithm which locates new solutions to widely used benchmarks. The performance of the metaheuristic is further improved by the development of an enhanced tuning approach using fitness clouds as behaviour models. The algorithm is shown to be further enhanced by taking advantage of multiprocessor environments, using threading techniques to parallelize the optimization workload. The work also shows quantum annealing applied successfully in an industrial setting to generate solutions to complex scheduling problems, results which created extra savings over an incumbent optimization technique. Components of the intellectual property rendered in this latter effort went on to secure a patent-protected status.

## Dedication

*If you want to be the best, it's all you need* - Roy Castle (RIP)

Without the stamina and advice of my primary supervisor, Alan Crispin, this work would never have been possible. Alan has been both a mentor and friend, and I will be always be grateful for his encouragement, support and criticism which are but a few contributions amongst numerous others he has made to my life and work. To his colleague, Keeley Crockett, I am also in debt. Her positivity, energetic exuberance and no-nonsense style were just the tonic the doctor/s ordered, helping to propel flagging efforts during the darkest nights of fevered rehashes and write-ups. (Added thanks for supervising, gratis, literal fuel on examination day in the form of quality coffee and sugary treats.)

Along the way to the completion of this work, I have met many fascinating people who, by virtue of their character, philosophical aspect, or advice, were of great inspiration. They are: Susan and Lenard Jubb, Dr Peter Torr, Dr Jim O'Shea, Dr Amy Khalfay, Dr Kristopher Welsh, Fiona Nightingale, and everyone at ServicePower especially Wendy Swann, Andrew Hill, Alan Smith, Frederic Turban, Andy Smith, and Tajinder Sandhu.

My mother, father, brother and sister, I thank for their continuing and steadfast tolerance of the terse or moody responses elicited by their good-natured but misapprehended questioning of this research. You have helped me to think of clearer explanations, to write to simplify, and never obfuscate.

During the periods of long studies and extended research, many changes to my life have occurred. I have now my daughters Phoebe and Harriet, a house and a respectable career. Without the support and direction of a spouse with infinite patience, none of these things nor the work contingent upon such, would have been even remotely possible. It is with love and immeasurable gratitude that I extend this most heartfelt dedication to my wife, Emma.

## Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Mr Alex Syrichas
6 January 2020

# Contents

# Glossary

**BKS** *Best Known Solution, Best Known Score.* The leading score given by an objective function for a particular problem instance and which is cited in literature.

**CVRP** *Capacitated Vehicle Routing Problem.* A variant of the Vehicle Routing Problem for which solutions must conform to a vehicle capacity constraint whilst servicing varying levels of customer demands.

**EA** *Evolutionary Algorithm.* A metaheuristic which takes inspiration from nature wherein facets of an optimization problem are analogous to biological adaptations - the fittest of which may evolve and be propagated through generations of breeding.

**ESPT** *Energy-based Scaled Parameter Tuning.* An improved method of predicting a good value of the temperature parameter for the algorithm FJ-QACVRP.

**FJ-QACVRP** *Fixed $J_\Gamma$ Quantum Annealing for the Capacitated Vehicle Routing Problem.* A derivative version of the quantum annealing metaheuristic which employs a constant field interaction strength, simplifying the prediction of other tunable parameters in order to solve vehicle routing problems.

**FQA** *Full Quantum Annealing.* In the domain of Field Service Scheduling, the name given to the metaheuristic which implements a complete Path-Integral Monte Carlo version of quantum annealing. This helps distinguish the algorithm from other derivatives in the same domain e.g. Quasi-Quantum Annealing.

**FSS** *Field Service Scheduling* A highly-constrained and dynamic industrial combinatorial optimization problem which shares features with the Technician Routing and Scheduling Problem, and Shop Scheduling Problems. A changeable backlog of jobs with a range of requirements must be assigned dynamically to a workforce which possess varying skills. A schedule must be provided to each worker which conforms to rosters and constraints such as location, inventory, service window, etc.

**GA** *Genetic Algorithm.* A metaheuristic whose mechanics are inspired by natural selection and genetics. Solutions are encoded as analogues of genomes which through successive generations mutate and hybridize, and so evolve to retain the features (genes) which are fittest whilst the weakest are suppressed.

**KTP** *Knowledge Transfer Partnership.* A UK government scheme, overseen by Innovate UK, with the aim of quickly transferring research and knowledge from academia to business. The partnership consists of a company, a university and a student who collaborate on a project to deliver innovations in areas such as services, technology, science, manufacturing or business processes.

**OR** *Operations Research.* A field of research which concerns itself with the use of analytical methods to improve decision making, planning and process management with the aim of producing savings through finding solutions to complex and usually intractable problems.

**PIMC** *Path-Integral Monte Carlo.* A statistical sampling method used to calculate quantum mechanical effects. As more sampling points (replicas) are included in the calculation, the accuracy of integral increases and will eventually approach the correct quantum result.

**PQA** *Parallelized Quantum Annealing.* The name given to variants of the quantum annealing metaheuristic which leverage multi-core processors and multithreaded environments, distributing the workload of Path-Integral Monte Carlo method to decrease the time taken to return a solution.

**PT Tuning** *Effective quantum temperature tuning.* A method of determining the temperature parameter for the algorithm QACVRP.

**QA** *Quantum Annealing.* A metaheuristic which can be viewed as the quantum counterpart of Simulated Annealing. The physical process of quantum tunneling is simulated using the Path-

Integral Monte Carlo method to provide an additional means by which the search performed by an optimizer can escape entrapment at local minima.

**QACVRP** *Quantum Annealing for the Capacitated Vehicle Routing Problem.* The quantum annealing metaheuristic which employed the Path-Integral Monte Carlo method as a single process to solve vehicle routing problems. The work performed in researching and implementing this algorithm is the foundation of this thesis.

**QQA** *Quasi-Quantum Annealing.* In the domain of Field Service Scheduling, the name given to the prototype version of the quantum annealing metaheuristic. Instead of employing the Path-Integral Monte Carlo method, the population of replicas is maintained using small random changes in interaction energy.

**SA** *Simulated Annealing.* A gradient-descent metaheuristic which employs a probabilistic acceptance criteria which allows a search to escape entrapment from local minimum. The mechanism behind this was inspired by the field of statistical thermodynamics, with the probability of accepting inferior solutions being dependent upon a temperature value which decreases during a analogue of the physical process of annealing.

**SASO** *Simulated Annealing for Service Optimization.* In the domain of Field Service Scheduling, the name given to a proprietary application of Simulated Annealing by ServicePower Technologies Plc.

**SSP** *Shop Scheduling Problem.* A broad category of combinatorial optimization problems which are concerned with finding schedules for a set of jobs to be performed upon a set of machines. The problems specify that solutions obey such constraints as precedence, and conform to machine restrictions such as no-wait and recirculation requirements, breakdowns and setup times. The objective function may combine such measures as tardiness, makespan (completion time of the last job), and maximum lateness (longest completed job).

**TRSP** *Technician Routing and Scheduling Problem.* An optimization problem with significance in several practical domains e.g. telephone network engineering. A set of jobs must be assigned to a workforce, providing constraint-conforming schedules to each individual, along with a route which minimizes the travel distance or time.

**TS** *Tabu Search.* A metaheuristic which employs a memory consisting of previous moves which are referred to with the expectation of reduced search repetition, increased diversification, and escaping entrapment at local minima. Each time a solution is added into memory or reused, it is assigned a timer during which it is marked as ‘taboo’. Once the timer has expired, the search may revisit the solution.

**TSP** *Traveling Salesman Problem.* A combinatorial optimization problem with the objective of assigning to an agent (salesperson) the shortest route which visits each of a collection of points (cities) once.

**VRP** *Vehicle Routing Problem.* A generalization of the Traveling Salesman Problem where many routes can be assigned to several or all of a set of agents (vehicles) to collectively form a solution with the shortest total distance.

# Chapter 1

# Introduction

## 1.1 Background

With contemporary society relying increasingly upon digital commerce, growing pressure is placed upon the infrastructure and activities which underpin businesses, especially those which involve online retailing and field servicing. Assets and functions such as vehicle fleets and service pools, storage and distribution hubs, manufacturing centers, and human resources are called upon to increase their efficiency, helping to sustain competitiveness as new products and service innovations are offered continually to customers, sold as improvements in convenience and thrift. In the past, external costs were rarely considered unless government regulations demanded the attention of businesses were turned towards environmental and social concerns. Recently however, governments around the world are collectively motivated to address environmental matters such as reducing greenhouse gas emissions, caused by the energy consuming activities of humankind. It has become more important than ever that efficiencies are found which not only increase the profitability of business but also reduce or eliminate environmental impacts. The field of operations research (OR) can be stated [1] to concern itself with matters such as these - applying analytical methods to improve decision making, planning and process management with the aim of producing savings through finding solutions to complex and usually intractable problems.

Consider a logistics business whose main activity is the delivery of products to geographically distributed customers using a fleet of vehicles. Within this process there may be many opportunities to find savings. For example, if the distances traveled by the fleet could be minimized, or the number of vehicles reduced, this would incur fuel savings and a reduction in $CO_2$ emissions. This simple restatement is the essence of the Vehicle Routing Problem (VRP), a combinatorial optimization problem perhaps first elucidated in 1959 [2]. VRP is a generalization of the famous Traveling Salesman Problem (TSP), first studied mathematically in the 1930s [3][4] to solve a bus routing problem. Whereas TSP is concerned with a single agent who joins up nodes to form a route, VRP consists of several agents (or a fleet) who each require a route of their own.

VRP can likewise be generalized as a scheduling problem such that the route describes a sequence of jobs instead of connecting customers, and where operatives (workers, technicians) replace the fleet. This collection of sequences would then form a work schedule. The reformulation of VRP instances into Shop Scheduling Problem (SSP) instances and vice-versa was accomplished [5], pointing to the relationship between these domains. However, the algorithms used to produce solutions were each tailored to their respective domains and showed indifferent performance upon the transformed problem instances. It was conjectured by the authors of this work that differences in the structural features of the problems were the cause of poor performance in the algorithms. Left unconsidered were the values and the tuning of the controlling parameters for the chosen algorithms.

The algorithms used to solve routing and scheduling problems are generally described in OR as optimizers, in that they search for better configurations by stepping through solution space, usually by making improvements to the current configuration. Improvements are judged by criteria defined in an objective function which often supplies a scalar value by which solutions can be ranked. The choice of objective function has a major effect upon the quality of the solution returned. It also plays a key role in the determination of the path followed through solution space. The space itself can affect the search. The space may be disjoint so that the best solutions are unreachable from the current, or the fitness landscape formed by the objective function contains steep-sided valleys which can entrap the search, or have shallow basins with little differentiation between solutions causing the search to stagnate. Watson [6] stresses the importance of understanding the behaviour of the search algorithm as it navigates the fitness landscape, proposing the use of cost models to capture this information.

The choice of neighbourhood operators is also a determinant of solution quality. A neighbourhood operator defines how a solution might change; it selects the possible directions that the search might take. Simple operators might move or swap the positions of singular elements in a solution to form a new configuration. Others might randomize a series. More complex and powerful operators exist which are domain-specific. In TSP and VRP there is the $\lambda$-optimal heuristic (Lin [7]), or 2-Opt move, which can 'unkink' crossed paths. For VRP and scheduling problems there is the cyclic $k$-transfers operation (Thompson and Psaraftis [8]) which transfers elements from many sequences or routes, simultaneously and in a cyclic fashion. The set of neighbourhood operators define what is termed the local search method. Bräysy and Gendreau [9] classify this as an approximation heuristic. It is important for the local search to contain enough different operators so that the solution space becomes non-disjointed. A typical design for an optimizer would also include a metaheuristic, which provides an overarching strategy to guide the local search scheme or to help it escape when it becomes stuck at minima in solution space. Bräysy and Gendreau [10] mention several strategies which the metaheuristic might include to restrict or reduce the complexity of search space. Garcia et al [11] state that in Tabu Search (TS) for VRP it is beneficial to prevent the neighbour operators from performing moves which involve customers who are too far apart. TS, by Glover [12], is a metaheuristic that allows inferior moves to be taken by the local search scheme whilst also attempting to prevent the search from revisiting configurations and needlessly repeating explorations of the same regions of solution space.

Perhaps the most famous metaheuristic is Simulated Annealing (SA) by Kirkpatrick [13] who applied it to TSP, and which was simultaneously proposed by Černý [14]. SA is a metaheuristic which takes inspiration from the thermodynamics of the process of annealing. At the core of SA is an implementation of the Metropolis-Hastings algorithm [15][16] which provides the thermodynamical statistics for the simulation of a system of particles. Attributes of the physical system are mapped to a combinatorial optimization problem. System states become feasible solutions; temperature becomes the controlling parameter of the acceptance criteria; energy becomes the objective cost. As SA is relatively simple to implement (being little more than a gradient descent algorithm incorporating a probabilistic mechanism which allows ascents) it has found widespread applications in fields as diverse as circuit design [17] and biology [18][19][20].

Also inspired by a physical process, Quantum Annealing (QA) uses simulated quantum mechanical tunneling effects to guide the local search. Brooke et al [21], and later Santoro et al [22] showed that because of tunneling, QA converged faster than SA. Battaglia et al [23] presented QA as a metaheuristic applied to satisfiability problems but had equivocal results, with SA performing better. Battaglia's conclusions suggest that field-cycling and better tuning would improve the results of QA. In the closely-related graph coloring problem, Titiloye and Crispin [24][25] had better results after expending effort upon tuning the controlling parameters.

The tuning of metaheuristics appears to be a neglected area of research. Often, the values of the controlling parameters of an algorithm are stated with little reason given for their selection. Some tuning schemes are performed by hand, and as such are incomplete or superficial. The tuning of SA is often left to the program itself via a cooling schedule [26], or adaptive mechanism [27]. Other algorithms can be employed to determine values of controlling parameters, techniques such as meta-optimization [28] and racing [29]. Meta-optimization techniques show some promise in tackling the automated tuning of metaheuristics with many parameters however, research in this field is limited to nature-inspired techniques such as Evolutionary Algorithms (Evolutionary Algorithm (EA)). The results of this kind of automated tuning may be indifferent owing to the choice of the problem/s [30], the choice of representative instance set, and mis-tuning of the meta-optimizer itself. Further, meta-optimization compounds one combinatorial optimization problem with another, and this restricts insight of the dynamic behaviour of the subject metaheuristic to views through the lens of the tuning metaheuristic. Finally, meta-optimization techniques like F-Racing [31] assume the interactions between parameters to be linear but this is not the case for many metaheuristics including QA, and so such techniques are unsuited for this research.

## 1.2 Optimization

Simply stated, an optimization process is a search for a particular configuration of an object within a space of all possible configurations. The search may be directed or random, and the space might be bounded (though large) or infinite.

*Consider a car; say, a Toyota Corolla. The Corolla is made up of some number of atoms; say, on the rough order of* $10^{29}$*. If you consider all possible ways to arrange* $10^{29}$ *atoms, only an infinitesimally tiny fraction of possible configurations would qualify as a car; if you picked one random configuration per Planck interval, many ages of the universe would pass before you hit on a wheeled wagon, let alone an internal combustion engine. Even restricting our attention to running vehicles, there is an astronomically huge design space of possible vehicles that could be composed of the same atoms as the Corolla, and most of them, from the perspective of a human user, won't work quite as well. We could take the parts in the Corolla's air conditioner, and mix them up in thousands of possible configurations; nearly all these configurations would result in a vehicle lower in our preference ordering, still recognizable as a car but lacking a working air conditioner. So there are many more configurations corresponding to nonvehicles, or vehicles lower in our preference ranking, than vehicles ranked greater than or equal to the Corolla. Similarly with the problem of planning, which also involves hitting tiny targets in a huge search space.* [32] - Eliezer Yudkowsky

Fortunately, most practical search problems are far less extreme than the one quoted above. In scheduling problems, the space of possible solutions though still vast, is far smaller. The search space can be further reduced by specifying that it contain viable schedules only, and clever use of structures inherent to the problem helps to carve the volume into promising regions which can then be intensively searched.

Many real-world problems fall into the mathematical field of *combinatorial optimization.* This topic seeks optimal solutions, ordinarily with a collection of constraints, from a finite set of possible solutions. Often the number of possible solutions is so large that it is impossible to exhaustively enumerate them such that the best can be found. For example, TSP can be defined as follows:

*An imaginary salesperson has to find a cyclic route which visits each of a number of cities only once whilst minimizing the total distance traveled.*

Although easily stated, it is not so easily solved. TSP belongs to a class of combinatorial optimization problems described as NP-hard, meaning that an algorithm which has to find an optimal solution by enumerating every route would run in exponential time i.e. the worst-case running time is bounded by $T(n) = O(2^n * n^2)$. The number of permutations of $n$ cities is given by the factorial $(n-1)!$ For values of $n \leq 14$, when the number of permutations ranges into the low billions, it is possible to employ a brute-force search to find the optimal route in a reasonable amount of time. However, practical (useful) formulations of the TSP such as those found in OR, like the VRP, are not so trivially bounded and business requirements place strict limits upon resources and computation time. A more realistic way to approach NP-hard problems is to use heuristic algorithms which, instead of attempting to find an absolute best solution, seek approximations or results 'good enough' within a permissible time. This relaxation of the constraint of optimality has allowed the production of a variety of algorithms and specially-tailored procedures which practically solve (and in some instances, to optimality) such problems. These algorithms may be categorized into three broad groups: construction, improvement and metaheuristic algorithms.

### 1.2.1 Construction, Improvement and Metaheuristic optimizers

*Construction* algorithms build a solution from nothing using simple, often greedy rules. These algorithms can be used to efficiently generate an upper bound by which to measure the performance of subsequent optimization methods. In VRP, construction heuristics work by creating routes by inserting customers (vertices) into partial routes or combining sub-routes taking into consideration capacities and costs. The Clarke and Wright saving method [33] is an example of a construction heuristic which starts by assuming that a single vehicle services a single customer and then constructs routes by calculating the saving that can be obtained by merging customers and servicing them with a single vehicle.

*Improvement* algorithms refine an existing solution by iteratively comparing (usually random) modifications to the current solution. Modifications are produced by defining a local search scheme consisting of a set of local or neighbourhood operators. New solutions are assigned a cost value provided by an objective function. This measure determines the replacement of inferior solutions with superior candidates. These kinds of local optimization methods and other descent procedures (hill-climbers) nearly always become 'stuck' in a local minimum. Unable to backtrack, they are fixed into the downhill path laid out by the selection of the last best known solution. The design of neighbourhood operators is key to the success of improvement algorithms - it is critical that the search be able to reach any (often enforced, feasible) solution in a single or series of changes. In summary, improvement heuristics seek to iteratively enhance a feasible solution (often initially generated by a construction heuristic) by replacing/swapping a set of elements locally.

*Metaheuristics* try to intelligently guide an improvement algorithm. These schemes are attractive because they can help the optimizer to escape from local minima by accepting inferior yet promising solutions from which to continue a local search. This assistance can take the form of backtracking either by maintaining a 'memory' of such solutions as in TS [34], or by making a stochastic selection as in EAs and SA.

A metaheuristic can be thought of as a top-level strategy which guides local improvement operators to find a global solution. Groër et al. [35] describe a library of local search heuristics for VRP.

### 1.2.2 Trajectory and population methods

A metaheuristic is described as a *trajectory* method [36] if it describes single path as it searches, moving sequentially from solution to solution through the space of possible solutions. TS, Late Acceptance Hill Climbing [37] and SA [13] are examples of trajectory methods.

A metaheuristic that maintains and updates many solutions simultaneously, is termed a *population* method. Members of the population may interact with one another, trading features which may enhance the fitness of the ensemble such as in Genetic Algorithm (GA)s and EAs. Others may maintain a ranked and perhaps diminishing population of best solutions such as in Ant Colony Optimization. At the simplest conception, a population method can be thought of as a multiple-trajectories method, which is able to explore many different paths simultaneously through solution space.

### 1.2.3 Energy-based metaheuristics

Metaheuristic inspired by physics, map energy minimization processes onto their optimization method. The objective function takes on the role of potential energy, and the sought-after optimal solution the ground state of a physical system. The process of searching the solution space becomes analogous the thermodynamical evolution of the chosen physical system.

## 1.3 Aims

The overriding goal of this research was to prove that QA, implemented using current technologies, is a practical and scalable alternative to SA. To achieve this, the key problem areas which would discourage the application of QA would need to be exposed and then addressed. Any deficiencies that were discovered during the research would likewise be made a priority to be addressed. The approach taken to this work was much like that of a software development production albeit one involving many experiments; if an unforeseen issue arises in design or outcome, a solution, workaround or alternative must be found lest the product fail.

An appropriate [30] problem domain with practical applications would need to be chosen. Given that QA was found to perform well for TSP [38], it would seem VRP is a good choice. VRP generalizes TSP, and has many variants with industrial applications. Additionally, since it is a well-researched domain, benchmarks exist with which to verify the behaviour and performance of the software. Thus the first goal was to produce a working version of QA for VRP with which to acquire empirical proof that it is at least as effective as SA in this domain.

Although it was known beforehand that a method of tuning would need to be formalized, it was found to be a critical issue during the development of QA for VRP. Algorithmic performance is dependent upon the structure of the problem's fitness landscape, an abstract topology formed by the combination of the set of all possible solutions (search space), an objective function, and move (neighbourhood) operators. Since a metaheuristic is the strategy for exploring this structure, the process of identifying precise values for the controlling parameters is of great importance. Metaheuristics possess one or more control parameters which must be chosen with great care before the algorithm is run. This is often done by beginning with best-guesses and then painstakingly refining over a series of experiments. Choosing the wrong parameters, even when out of adjustment by a fraction of a percent, can mean at best, sub-optimal results and at worst, the algorithm will never converge. With so many interdependent controlling variables, tuning the QA algorithm was difficult and slow. Hence the next goal was to simplify and perhaps automate the configuration of the algorithm. Results needed to show their worth compared to the resources expended in tuning the algorithm. Competitive or new best solutions were sought.

As the tuning research expanded the number of benchmarks used and their complexity increased, running time issues became a bottleneck upon delivery of experimental results. In industrial settings, it would be an unacceptable situation to be waiting upon schedules to be produced. Hence improving the runtime performance became an important aim, with the main direction being to take advantage of modern computing platforms and scalable architectures.

Midway through these goals, another objective arose - an opportunity to apply QA and supplant SA in an industrial setting. ServicePower Technologies Plc, requiring improvements in schedule quality for actual workforces, sought a partnership to develop a practical application of quantum annealing.

## 1.4 Overview

This thesis is divided into four parts, in approximately chronological order. The time line starts with the foundational work for the quantum annealing algorithm, and ends with an industrial application having proven uses.

- **Chapter 2: Quantum Annealing Algorithm for the Vehicle Routing Problem** This chapter presents the first application of quantum annealing in the domain of the Vehicle Routing Problem. It demonstrates a somewhat effective method for tuning the algorithm, and presents an empirical comparison with Simulated Annealing using well known benchmarks. The comparisons are on the basis of solution quality and success rate as a percentage.
- **Chapter 3: Tuning QA for the Vehicle Routing Problem** This chapter documents the effort to improve the tuning methodology. Given the number of controlling parameters that quantum annealing possesses and complexities which arise because of their interde-

pendence, the work attempts to reduce their number and formalize a method of predicting the correct temperature value. The work also extends the empirical study to include larger benchmarks and support solving the distance-constrained variant of the problem.

- **Chapter 4: Quantum Annealing Algorithm: Enhancements and Variations** This chapter shows how the workload of the algorithm may be distributed such that the running time is improved. Designs are shown for two threading models, along with program pseudo-code, and empirical comparisons with the previous algorithms. Results are compared in terms of the wall-clock time taken for the algorithms to find equivalent (optimal) solutions for well-known benchmarks.

- **Chapter 5: The Industrial Domain of Field Service Scheduling** This chapter documents the application of quantum annealing in an industrial scheduling domain. Empirical studies were performed upon datasets drawn from real life scenarios (client names redacted), the results compared with those of an existing product built upon Simulated Annealing. The quantum annealing algorithm goes through several revisions resulting in a multithreaded version delivering results at high speed. It should be noted that some of the work is intellectual property owned by a private company, and so could not be reproduced here.

## 1.5 Contributions

1. **First application of QA to VRP** So far as is known, this work is the inaugural application of QA to the vehicle routing problem. Specifically, a Path-Integral Monte Carlo (PIMC) method is implemented and quantum fluctuations are simulated in order to solve the capacitated variant of the problem which is mapped on to an Ising model.

2. **Encoding of VRP as a spin matrix** The mapping of the problem onto an Ising model is at the core of simulating quantum effects upon a classical computer. The research presents an intuitive method for the encoding of the specifics of the problem as a set of $\{+1, -1\}$ spin values. Consequently, memory consumption was minimized, (resulting in fewer misses of the cpu data cache and a reduction in latency) and faster execution of the program.

3. **Energy-based Scaled Parameter Tuning method** The dynamic cost model [6] which described the run time behaviour of QA provided the key insight which was needed to develop a new tuning method entitled Energy-based Scaled Parameter Tuning (ESPT). The fitness cloud representation of the behaviour showed that the topology of the fitness landscapes (within classes of instances) was similar enough for the predicted temperature value to succumb to tuning by a scaling factor. As well as simplifying the task of hand tuning, the steps of this new method can be captured as instructions for a script or program, potentially culminating in automated tuning process. Fitness clouds have previously been used in the study of evolutionary algorithms to estimate the difficulty of problem instances. Their use as behaviour models in the study of energy-based metaheuristics such as QA, as far as is known, has never been tried before.

4. **New best solutions in benchmarks** Although the competitive testing paradigm is non-productive [6] in research such as this, it should be pointed out that these new best solutions were generated as a consequence of studying the fitness landscape, the problem domain, and the QA metaheuristic. Acknowledging this achievement will hopefully encourage others to work to understand why their algorithms perform well.

5. **New parallelized algorithm for QA** After the issue of the difficulty of tuning QA was addressed, it was natural and desirable to enhance the runtime performance of the algorithm. If QA is to become a solid choice of metaheuristic to supplant SA, then all key objections to deploying it must be quashed. It should be noted that unlocking the performance of QA strictly requires a platform capable of hosting a large population of threads. The effort to improve the performance of QA is the subject of an article [39] under review and awaiting publication.

6. **First practical application of QA in industry** To the best of our knowledge, this is the first time QA has been used in an commercial role. That it is able to solve complex scheduling problems as well as idealized academic benchmarks is a notable achievement. Section 1.5.2 documents the publicity generated by the research efforts expended in the Field Service Scheduling (FSS) domain.

7. **Three new QA-based algorithms for Field Service Scheduling** The path to the completion of a new scalable optimizer for FSS was punctuated with three milestones - Quasi-Quantum annealing (QQA), Full Quantum Annealing (FQA), and Parallelized Quantum Annealing (PQA). QQA, a derived algorithm developed first, was a vital step which proved that a population-based metaheuristic was a good fit for the industrial domain. As a proof of concept, it was shown to also be a minimum viable product since it could optimize nearly as well as FQA.

8. **Patent approved for spin encoding** The design for the spin encoding of a FSS problem was awarded a patent [40] by the US Patent Office. This underscores the novelty of matrix arrangement whilst also accruing valuable intellectual property and marketing opportunities for ServicePower.

### 1.5.1 Research outputs from this work

1. **Conference paper with presentation**

   Alan Crispin and Alex Syrichas. Quantum annealing algorithm for vehicle scheduling. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, October 2013. doi: 10.1109/smc.2013.601. URL `https://doi.org/10.1109/smc.2013.601`

2. **Journal article**

   A. Syrichas and A. Crispin. Large-scale vehicle routing problems: Quantum annealing, tunings and results. *Computers & Operations Research*, 87:52–62, November 2017. doi: 10.1016/j.cor.2017.05.014. URL `https://doi.org/10.1016/j.cor.2017.05.014`

3. **Journal article, submitted and under review**

   A. Syrichas and A. Crispin. A parallelized quantum annealing algorithm for vehicle routing problems. *Operations Research Perspectives*, 2019. (Under review)

4. **US Patent awarded**

   A. Syrichas and A. Crispin. Encoding of a schedule into a binary structure, 2017. US Patent 9,841,990

### 1.5.2 News and public engagement articles

The following is a selection of news and intrest articles garnered by this research.

1. **Manchester Metropolitan University Press Release (2018)**

   Manchester Metropolitan University Press Release. Quantum physics-based algorithm helps companies better manage mobile workforces: University's patented code powers field service management software, 2018. URL `http://www2.mmu.ac.uk/news-and-events/news/story/7359`. Accessed 23 June 2019

2. **Manchester Metropolitan University Press Release (2017)**

   Manchester Metropolitan University Press Release. Servicepower case study: State of the art software helps industry improve vehicle logistics by up to 10%, 2017. URL `https://www2.mmu.ac.uk/business/success-stories/detail/service-power.php`. Accessed 23 June 2019

3. **Electronics Times (2017)**

   Electronic Times: Software Simulations Now. Quantum computing uses standard hardware, 2017. URL `https://www.eetimes.com/document.asp?doc_id=1331768`. Accessed 23 June 2019

4. **Field Service News (2017)**

   Field Service News. Is quantum computing the future of customer experience and service optimisation?, 2017. URL `http://fieldservicenews.com/quantum-computing-future-customer-experience-service-optimisation`. Accessed 23 June 2019

5. **Manchester Metropolitan University Press Release (2016)**

   Manchester Metropolitan University Press Release. Quantum computing solves logistics conundrum. algorithm helps lorries deliver goods efficiently, 2016. URL `http://www.mmu.ac.uk/news/news-items/4204`. Accessed 23 June 2019

6. **ServicePower Whitepaper (2016)**

   ServicePower Inc. David and goliath: Redefining field service with quantum annealing, 2016.

URL `https://www.servicepower.com/understanding-quantum-annealing-awareness-lp`. Accessed 23 June 2019

7. **World News Press Release (2016)**

   World News. Quantum computing solves logistics conundrum, 2016. URL `http://article.wn.com/view/2016/03/10/Quantum_computing_solves_logistics_conundrum_Manchester_Metr`. Accessed 23 June 2019

8. **ServicePower Press Release - Patents applications (2015)**

   ServicePower Inc. Servicepower applies for various quantum annealing patents, 2015. URL `https://www.servicepower.com/blog/servicepower-applies-for-various-quantum-annealing-patents`. Accessed 23 June 2019

# Chapter 2

# Quantum Annealing Algorithm for the Vehicle Routing Problem

## 2.1 Introduction

This chapter proposes a new strategy for solving the Capacitated Vehicle Routing Problem (CVRP) using a Quantum Annealing (QA) algorithm. CVRP is a variant of VRP being characterized by capacitated vehicles which contain goods up to a certain maximum capacity. QA is a metaheuristic that uses quantum tunneling in the annealing process. A spin encoding scheme is discussed and devised for solving CVRP with a QA algorithm and an empirical approach for tuning parameters. The effectiveness of QA is studied in comparison with best known solutions for a range of benchmark instances

CVRP is a variant of VRP [2] in which all vehicles have the same capacity. It has important industrial applications as optimal solutions enable transportation costs to be reduced and customer service levels to be improved. The underlying structure of the CVRP is an undirected graph $G = (V, E)$, where $V = \{v_0, v_1, ...v_n\}$ is a vertex set and $E = \{(v_i, v_j)|v_i, v_j \in V, i < j\}$ is an edge set. The restriction $i < j$ ensures that the problem is symmetric i.e. the distance between two vertices is identical in both directions. Vertex $v_0$ is the depot from where a fleet of $m(m \geq 1)$ identical vehicles, which may contain goods up to a certain maximum capacity $Q$, must serve exactly $n$ customers represented by the set of $n$ vertices $\{v_1, v_2, ...v_n\}$. A nonnegative cost $d_{ij}$ can be defined in terms of a distance or time between vertices (customers) $v_i$ and $v_j$. Each customer has a non-negative demand for goods $q_i$. The cost of the problem solution is the sum of the costs of its routes.

Rose and Macready [51] describe how QA can be used to solve optimization problems. In order to implement QA on a classical computer, it is necessary to use a stochastic variant such as PIMC, see Titiloye and Crispin [24]. This requires the development of a Hamiltonian consisting of the sum of two terms $H_p$ and $H_k$ where $H_p$ represents the classical potential energy of a given problem configuration and $H_k$ is a suitable kinetic energy term, devised for the problem, to provide quantum fluctuations during a Monte Carlo cycle. The kinetic energy term consists of a population of candidate solutions, encoded as spin matrices, with each member of the population known as a replica. In essence, QA is a population heuristic as it uses a number of current solutions (replicas) and combines them to generate new solutions. Consequently, Monte Carlo QA shares features with both Simulated Annealing (SA) and an Evolutionary Algorithm (EA). However, whereas SA attempts to overcome barriers thermally, Monte Carlo QA simulates quantum tunneling between the replicas to escape barriers to the ground state. The analogy with EA is drawn as both are based on the interaction of member solutions in a population.

The Path-Integral Monte Carlo Quantum Annealing algorithm has been applied with success to the Traveling Salesman Problem (TSP) [38]. An undirected TSP tour is represented in terms of Boolean variables by defining a connection matrix from which Ising spin variables are defined. A 2-opt move is used to attempt to improve a tour configuration in the Monte Carlo algorithm. A 2-opt move is a local improvement operator which tries to improve a tour configuration by removing two edges in the tour and replacing them with two new edges. In essence, TSP is a simplified variant of the VRP where the goal is to find the shortest possible tour, in terms of total distance, which visits each of the cities exactly once. This corresponds to a VRP where there is only one vehicle, no capacity restrictions, and the cost is only dependent on, and directly proportional to, the distance. The similarity between TSP and VRP suggests that a QA heuristic could be established for vehicle scheduling and [38] identifies the use of constrained Boolean variables to represent the problem as a key factor to success.

This section is organized as follows. Section 2.2 describes the quantum annealing scheme for the CVRP. Section 2.3 provides pseudo-code for the PIMC quantum annealing algorithm and details of parameter tuning are given in section 2.4. In section 2.5 results from computational experiments on CVRP benchmark data sets are presented and compared against a SA algorithm. Section 2.6 analyzes the QA algorithm. Concluding remarks are given in section 2.7.

## 2.2 QA Scheme for CVRP

In the quantum PIMC approach, the statistical behavior of the quantum spin model is approximated to a classical simulation model by using a Suzuki-Trotter transformation [52]. With PIMC, the quantum system is mapped onto a classical model consisting of $P$ ferromagnetically coupled spin matrices or replicas. The total Hamiltonian energy $H$ can be written as [23]:

$$H = \frac{H_p}{P} - J_\Gamma \Delta H_k \tag{2.1}$$

Here $H_p$ is the potential energy summed over the $P$ replicas so that $H_p/P$ is the average value and $J_\Gamma$ is the transverse ferromagnetic coupling term given by:

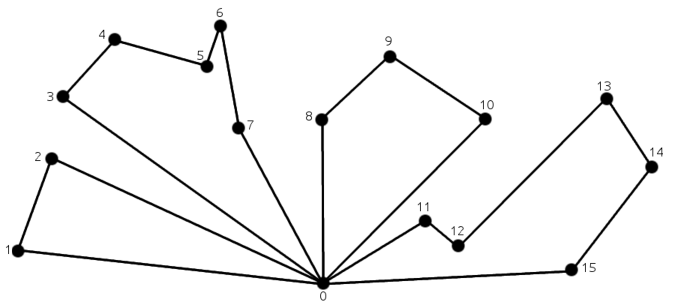
$$J_\Gamma = \frac{-T}{2} \ln \tanh \left( \frac{\mathit{\Gamma}}{PT} \right) \tag{2.2}$$

In equation (2.2), $T$ is the temperature and $\mathit{\Gamma}$ the tunneling field strength parameter. The term $PT$ is called the effective quantum temperature [23]. The change in kinetic energy term $\Delta H_k$ provides a disturbance during a Monte Carlo iteration cycle for escaping local minima and is calculated from replica interactions. To do this a set of $P$ spin matrices has to be defined with each matrix randomly initialized. The idea is that the change in kinetic energy $\Delta H_k$ should decrease to zero over time as the replicas become similar so that $H$ tends to the average potential energy value. The strength of $J_\Gamma$ determines the importance of the $\Delta H_k$ term relative to the potential energy term. Notice that $J_\Gamma$ is positive for ferromagnetic interaction.

To apply QA to a problem requires finding an intuitive way to encode the problem in terms of a set of spin matrices or replicas. The spin matrix $\sigma_{ij}$ is defined from a connectivity matrix which describes all routes in a VRP schedule. Figure 2.1 shows an example of the spin encoding scheme. In this example there are fifteen customers (vertices) which are served from a depot ($v_0$)

by four routes, one for each vehicle. Rows $i$ and columns $j$ are labeled 0 to 15 and represent the vertices in the network. Each cell represents a connection between two vertices and $\sigma_{ij} = 1$ is defined if vertex $i$ is connected to vertex $j$ in a route and $\sigma_{ij} = 0$ otherwise. In general, the size $N$ of the spin matrix is equal to the number of vertices in the problem and all diagonal elements are zero. As connections in the network are bi-directional the spin matrix is symmetric. The potential energy $H_p$ for the system can be defined as:

$$H_p = \sum_{P}\sum_{i,j} d_{P,i,j}\,\sigma_{P,i,j} \tag{2.3}$$



| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Hex** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 8D8E |
| **1** | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0005 |
| **2** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0003 |
| **3** | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0011 |
| **4** | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0028 |
| **5** | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0050 |
| **6** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00A0 |
| **7** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0041 |
| **8** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0201 |
| **9** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0500 |
| **10** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0201 |
| **11** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1001 |
| **12** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 2800 |
| **13** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 5000 |
| **14** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | A000 |
| **15** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 4001 |

Figure 2.1: Example of routes encoded as a spin matrix. Customer-customer connections are represented using single bits. The matrix is encoded row-wise to form hexadecimal (Hex) words which are stored in memory as an array.

In equation (2.3), $d_{z,i,j}$ is the distance between vertex $i$ and vertex $j$ in the $z^{th}$ replica and

$< i, j >$ is used to signify counting each connection only once [38]. In any one replica the classical potential energy is the total length of all routes. It is because the potential energy is calculated over a set of $P$ replicas that $H_p$ is divided by $P$ in equation (2.1) to determine the average value. The $H_p/P$ term represents the solution sought, provided vehicle capacity is treated as a constraint so that a customer cannot be added to a route if the vehicle capacity $Q$ is exceeded. The solution space can be repeatedly sampled until a route configuration is found which does not violate vehicle capacity constraints and this is the approach used in this work.

To calculate a kinetic energy each spin matrix (replica) needs to know the state of its neighbours. The kinetic energy of the entire system is calculated as:

$$H_k = J_\Gamma \sum_P \sum_i \sigma_{P-1,i} \sigma_{P,i} \sigma_{P+1,i} \tag{2.4}$$

Here the first term of equation (2.4) is the sum of products of the current replica with its previous replica over all $P$ replicas where $z$ is the replica index. Likewise, the second term is the sum of products of the current replica and its next replica over all $P$ replicas. The first and last replicas are connected to make a circular list so that they can be provided with previous and next neighbours. Like [38], no use is made of single spin-flips. Instead, local VRP neighbourhood operators are used when attempting to improve a configuration to ensure that a valid network results and a constrained Ising model is maintained.

## 2.3 Quantum Annealing Algorithm

The pseudo-code for the Path-Integral Monte Carlo QA algorithm for CVRP is shown as figure 2.2.

**Variables**: $P$: number of replicas, $T$: temperature, $\mathit{\Gamma}$: magnetic field, $\Delta\mathit{\Gamma}$: decrement value of magnetic field, $M_C$ : number of Monte Carlo steps, $N$: set of neighbourhood operators, $n$: chosen operator, $S$: set of replicas, $S_{best}$ : best solution, $s'$: candidate solution, $z$: index of current replica.

```
1:  $S \leftarrow$ circular list of feasible randomized solutions
2:  $S_{best} \leftarrow S_0$
3:  **while** $M_C > 0$ **do**
4:      $J_\Gamma \leftarrow (-T/2)\ln(\tanh(\mathit{\Gamma}/PT))$
5:      $z \leftarrow 0$
6:      **while** $z < P$ **do**
7:          randomly choose $n \in N$
8:          $s' \leftarrow n(S_z)$
9:          $\Delta H_p \leftarrow H_p(s') - H_p(S_z)$
10:         $\Delta H_k \leftarrow \text{Interact}(S_{z-1}, s', S_{z+1}) - \text{Interact}(S_{z-1}, S_z, S_{z+1})$
11:         $\Delta H \leftarrow (\Delta H_p/P) + J_\Gamma \Delta H_k$
12:         **if** $((\Delta H_p <= 0)$ **or** $(\Delta H <= 0))$ **then**
13:             $S_z \leftarrow s'$
14:         **else if** $\exp(-\Delta H/T) > \text{random}(0, 1)$ **then**
15:             $S_z \leftarrow s'$
16:         **end if**
17:         **if** $H_p(S_z) < H_p(S_{best})$ **then**
18:             $S_{best} \leftarrow S_z$
19:         **end if**
20:         $z \leftarrow z + 1$
21:     **end while**
22:     $\mathit{\Gamma} \leftarrow \mathit{\Gamma} + \Delta\mathit{\Gamma}$
23:     $M_C \leftarrow M_C - 1$
24: **end while**
25: return $S_{best}$
```

Figure 2.2: Quantum Annealing for Capacitated Vehicle Routing Problems.

Quantum Annealing for the Capacitated Vehicle Routing Problem (QACVRP) requires the following input parameters: the number of replicas $P$, the temperature $T$, gamma $\mathit{\Gamma}$, the rate of decrease of gamma $\Delta\mathit{\Gamma}$, the maximum number of Monte Carlo steps $M_C$ and a set of $N$ neighbourhood operators (insert, swap, 2-opt, cross, scramble, string-insert and 2-opt-Star). The insert operator moves one customer from one route to another route. The swap operator randomly chooses two customers from different routes and exchanges them. The 2-opt operator randomly selects two non-adjacent edges of a single route, and then reverses the connection order of the customers between the outer endpoints. The cross operator randomly chooses two sequences from different routes and swaps them. The scramble operator randomly chooses two customers from one route and shuffles the connection order between them. The string-insert operator removes a sequence of connections from one route and places it into a different route. The 2-opt-Star operator exchanges random length end portions of two routes, preserving the order of connections between customers. For a further discussion on VRP neighbourhood operators see Bräysy and Gendreau [9].

On line 1, $S$ is computed S as an initial set of spin matrix replicas forming a circular list. The initial population of $P$ spin replicas is generated using a random constructive algorithm so that a diversified set of spin matrix replicas is built. The construction algorithm randomly chooses a number of routes and then randomly selects and inserts vertices, one at a time, into partial routes until a feasible solution is obtained taking capacity constraints into account. On line 2 the best spin matrix replica solution is initialized. On line 3 the Monte Carlo loop is started using the Monte Carlo maximum step value $M_C$ . $J_\Gamma$ is calculated on line 4 and on line 5 the initial value of the index $z$ is set to be zero, the first replica. On line 6 the replica loop is started. On line 7 a local operator is randomly chosen from the set of operators (insert, swap, 2-opt, cross, scramble, string-insert and 2-opt-Star) which is then applied (line 8) to create a modified replica solution $s'$. On line 9 the difference in potential energy is found using the modified replica solution $s'$ and the unmodified replica $S_z$. On line 10 the difference in spin products is calculated, again using both the modified replica solution $s'$ and the unmodified replica $S_z$ to determine if the kinetic energy has changed. The total energy change is calculated on line 11.

On line 12 if either the change in potential energy $\Delta H_p$ or the change in total energy $\Delta H$ has decreased the modified solution is assigned to $S_z$ (line 13). If the cost value of the modified replica solution is worse than the previous value it can still be accepted (line 14) using the criteria: $exp(-\Delta H/T) > random(0, 1)$. That is, the value $exp(-\Delta H/T)$ is compared to a random number between 0 and 1 and, if greater, the modified solution $s'$ is assigned to $S_z$. The acceptance of a worse solution (Metropolis criteria) is controlled by the change in total energy $\Delta H$ and the temperature, providing a mechanism for accepting worse solutions with the aim of escaping local minima.

On line 17 the potential energy of the current replica solution $S_z$ is calculated and if it is better than the current best potential energy, then $S_z$ is assigned as the current best solution (line 18). Line 20 increments the current replica index $z$ and ends the inner replica calculation loop. Line 22 provides an option to decrement gamma. Line 23 decrements the Monte Carlo step value ending the Monte Carlo step loop. Line 25 returns the best replica solution found.

Notice that the algorithm calculates changes in energy rather than calculating the Hamiltonian energy for the entire system at every iteration which would be computationally expensive especially for non-trivial problem instances where $P$ is large. $\Delta H_p$ is calculated simply as the difference between the potential energy of the candidate $s'$ and the previously calculated potential energy value. $\Delta H_k$ is calculated as the difference between the spin product of the candidate $s'$ and the previously calculated spin product state.

**Variables**: $S_{z-1}$, $S_z$, $S_{z+1}$: spin matrices belonging to previous, current, and next replicas, $i, j$: matrix dimensions, $E$: interaction energy.

```
1: $E \leftarrow \sum_{i,j} S_{z,i,j} S_{z-1,i,j}$
2: $E \leftarrow E + \sum_{i,j} S_{z,i,j} S_{z+1,i,j}$
3: return $E$
```

Figure 2.3: Interact: calculates the interaction energy between neighbouring replicas

Figure 2.3 shows the pseudo-code for the Interact($S_{z-1}$, $S_z$, $S_{z+1}$) function used when calculating the kinetic energy difference.

The QACVRP algorithm described in figures 2.2 and 2.3 has been implemented in C++ using the Qt framework and compiled using GNU GCC on a PC running a Linux operating system (see appendices A.1 and A.3). The computer hardware set-up consisted of two cores and independent simulations were run on each of the cores to obtain the experimental results discussed in section 2.5.

With the PIMC scheme, the accuracy of the quantum simulation increases for larger values of $P$ but using larger $P$ requires more memory and computation [23]. However, the VRP spin encoding system for QACVRP is extremely memory efficient allowing the use of a large value for $P$ (e.g. $P = 40$) in the simulations that follow. The chosen format of the spin matrix (see Figure 2.1) optimizes the memory storage for a VRP network. Since connections between customers can be represented in a single bit, memory usage can be reduced by storing each row of the matrix in a bit-stream represented as a hexadecimal number in Figure 2.1. On a 64-bit based system each memory word can store up to 64 connections; so for example, a problem instance of 128 customers, with a matrix of (128x128) can be stored with 2x128 memory words. This allows the change in kinetic energy between matrices to be calculated efficiently. Sampling the bit-streams in whole memory words and using logical rather than arithmetic operations allow 64 spin products to be calculated together.

## 2.4 Parameter Tuning

The QA algorithm in figures 2.2 and 2.3 requires that the parameters $P$, $T$, $\mathit{\Gamma}$ and $\Delta\mathit{\Gamma}$ be tuned to ensure robust and high performance. To simplify tuning and to make a quantum annealing algorithm more robust, the rate of decrease of gamma can be set to zero, provided gamma itself is carefully chosen for the problem instance [25]. With this approach only three parameters ($P$, $T$, $\mathit{\Gamma}$) need to be tuned for good performance. This work devises a methodology for tuning the QA parameters based on empirically measuring the success rate of obtaining a known CVRP optimal value as the parameters are systematically changed. The problem instance p-n101-k4 (101 nodes and 4 routes) was chosen as the tuning instance which has a known optimal value of 681:4 (best length of 681 with 4 routes). The objective was to determine a set of $P$, $T$, $\mathit{\Gamma}$ values which result in a 100% success rate. A series of computational experiments was performed to enable the success rate to be plotted against an effective quantum temperature $PT$ for different values of $P$. That is, the effective quantum temperature was changed by keeping $P$ fixed and increasing $T$. In these experiments the Monte Carlo maximum steps value $M_C$ was set to $5{\times}10^6$ and $\mathit{\Gamma} = 3$. The computational experiments showed that the success rate of finding the best solution for p-n101-k4 increases with the number of replicas used in the simulation. For $P = 10$ the success rate peaks at $PT = 0.9$ with a value of 77%. For $P = 20$ the success rate peaks at $PT = 0.9$ with a value of 91%. When $P = 40$, the success rate is 100% in the range $0.9 < PT < 1$, as shown in Figure 2.4.

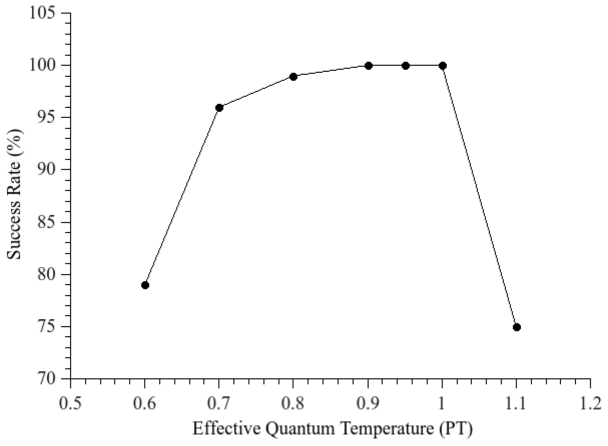


Figure 2.4: Tuning of the effective quantum temperature for the CVRP instance P-n101-k4. As temperature $T$ is increased, the success rate (percentage of optimal results) is improved until a cut-off point is reached. The number of replicas $P$ is held constant to simplify the process of tuning. The best values for $T$ and $P$ are found where the success rate peaks.

In all cases the success rate steadily increases as $PT$ is increased and after reaching a peak value it then declines steeply. It is clear by inspecting Figure 2.4 that $PT = 0.9$ results in 100% success rate and so $P = 40$, $T = 22.5{\times}10^{-3}$, $\varGamma = 3$ with $M_C = 5{\times}10^6$ have been used as the base set of tuned parameters for this study.

## 2.5 Experimental Results

Table 2.1 shows results of experiments for QACVRP for a set of well-known benchmark instances. Column 1 shows the names of the problem instances, and column 2 the best known solutions (BKS) found using the branch and cut method [53]. Column 3 contains the results obtained from the QACVRP algorithm for 100 independent runs for statistical significance and is subdivided into the best value obtained by the QACVRP algorithm, the wall clock time and the percentage success rate of obtaining the best known solution for a particular instance. In all instances, except Mn121-k7 and F-n135-k7 (discussed below), the parameters $P = 40$, $T = 22.5{\times}10^{-3}$, $\varGamma = 3$ with $M_C = 5{\times}10^6$ were used. Column 4 contains the results obtained for a fixed temperature SA for 100 independent runs using the same local operators as QA on the same hardware configuration. To allow for a fair comparison both algorithms were run for the same number of iterations (i.e.

the SA was run for $200{\times}10^6$ iterations which is equivalent to QA with a 40 replica inner loop and $5{\times}10^6$ outer Monte Carlo loop). The SA temperature was initially tuned for the p-n101-k4 instance (i.e. $T = 1$) and then, if necessary, optimized for other instances to obtain the best possible results.

Table 2.1: Computational results for benchmarks.

| **Instance** | **Best known** | **QA** | | | **SA** | | |
|---|---|---|---|---|---|---|---|
| | | **Score** | **Time** | **Success %** | **Score** | **Time** | **Success %** |
| P-n40-k5 | 458:05 | 458:05 | 00:00:50 | 100 | 458:05 | 00:00:26 | 100 |
| P-n45-k5 | 510:05 | 510:05 | 00:01:53 | 100 | 510:05 | 00:36:51 | 93 |
| P-n50-k7 | 554:07 | 554:07 | 00:09:14 | 100 | 554:07 | 00:04:45 | 100 |
| P-n50-k10 | 696:10 | 696:10 | 08:27:14 | 63 | 696:10 | 03:57:49 | 28 |
| P-n51-k10 | 741:10 | 741:10 | 01:12:35 | 100 | 741:10 | 02:39:49 | 50 |
| P-n55-k7 | 568:07 | 568:07 | 01:11:40 | 100 | 568:07 | 00:30:58 | 99 |
| P-n55-k10 | 694:10 | 694:10 | 10:14:24 | 35 | 694:10 | 02:26:43 | 31 |
| P-n60-k10 | 744:10 | 744:10 | 01:07:38 | 100 | 744:10 | 01:27:26 | 73 |
| P-n60-k15 | 968:15 | 968:15 | 08:48:46 | 79 | 968:15 | 03:12:11 | 79 |
| P-n65-k10 | 792:10 | 792:10 | 01:01:47 | 100 | 792:10 | 00:30:04 | 100 |
| P-n70-k10 | 827:10 | 827:10 | 13:23:50 | 78 | 827:10 | 05:25:02 | 61 |
| P-n76-k4 | 593:04 | 593:04 | 14:50:56 | 52 | 593:04 | 03:50:16 | 44 |
| P-n76-k5 | 627:05 | 627:05 | 10:16:25 | 87 | 627:05 | 04:38:44 | 22 |
| P-n101-k4 | 681:04 | 681:04 | 07:56:20 | 100 | 681:04 | 00:46:35 | 97 |
| B-n50-k8 | 1312:08 | 1312:08 | 01:34:54 | 100 | 1312:08 | 00:10:10 | 100 |
| B-n52-k7 | 747:07 | 747:07 | 00:09:06 | 100 | 747:07 | 00:00:46 | 100 |
| B-n56-k7 | 707:07 | 707:07 | 00:20:34 | 100 | 707:07 | 00:01:55 | 100 |
| B-n57-k9 | 1598:09 | 1598:09 | 00:40:50 | 100 | 1598:09 | 00:04:31 | 100 |
| B-n63-k10 | 1496:10 | 1496:10 | 12:49:30 | 26 | 1496:10 | 02:37:01 | 25 |
| B-n64-k9 | 861:09 | 861:09 | 01:04:16 | 100 | 861:09 | 00:17:23 | 100 |
| B-n66-k9 | 1316:09 | 1316:09 | 08:56:25 | 91 | 1316:09 | 02:45:03 | 44 |
| B-n67-k10 | 1032:10 | 1032:10 | 15:42:00 | 42 | 1032:10 | 01:14:48 | 88 |
| B-n68-k9 | 1272:09 | 1272:09 | 11:30:52 | 69 | 1272:09 | 01:29:32 | 87 |
| B-n78-k10 | 1221:10 | 1221:10 | 07:10:29 | 97 | 1221:10 | 00:36:48 | 99 |
| M-n121-k7[a] | 1034:07 | 1034:07 | 03:08:51 | 90 | 1034:07 | 04:57:44 | 76 |
| F-n135-k7[b] | 1162:07 | 1162:07 | 00:45:25 | 11 | 1162:07 | 02:59:43 | 4 |

[a] $P = 50$, $T = 12{\times}10^{-3}$, $\mathit{\Gamma} = 3$, $M_C = 5{\times}10^6$
[b] $P = 50$, $T = 6{\times}10^{-3}$, $\mathit{\Gamma} = 3$, $M_C = 10{\times}10^6$

As can be seen from Table 2.1, the QACVRP algorithm matches or outperforms SA in terms of success rate for every case except three of the Augerat set B instances. The QACVRP results are also competitive compared to those reported in the literature for other population heuristics. For example with p-n50-k7, QA finds the BKS with a success rate of 100% whereas the genetic algorithm result [54] differs from the BKS by up to 7.27%. For p-n101-k4, QA again has a success rate of 100% whereas the result for the memetic algorithm [55] shows best and worst values.

With more difficult instances, for example M-n121-k7 and F-n135-k7, it is necessary to re-tune the $P$, $T$ and $\Gamma$ parameters. The problem instance F-n135-k7, which represents a day of grocery deliveries, is particularly difficult for VRP algorithms [56]. With $PT$ set to 0.3, (i.e. a temperature value of 0.006 with the replica number fixed at 50) and $\Gamma$ set to 3, QA was able to obtain the optimal value. The success rate of obtaining the optimal value was 5% when the number of Monte Carlo steps was set to $5\times10^6$. Increasing the Monte Carlo steps to $7.5\times10^6$ resulted in a success rate of 10%. The best success rate obtained was 11% with the Monte Carlo steps set to $10\times10^6$. Increasing the maximum number of Monte Carlo steps extends the simulation time.

Overall, QA reaches a maximum success rate in 50% of the instances versus 31% for SA, and outperforms SA 89% of the time. However, the success rate can swing as low as 11% which indicates that the values of the parameter set are not universally ideal.

## 2.6 Analysis of QA

The QACVRP algorithm requires that a maximum number of Monte Carlo steps is defined. Consequently, it was investigated how the best solution averaged over 100 runs changes as the maximum number of Monte Carlo steps is increased. The results for p-n101-k4 with $P = 40$, $T = 22.5\times10^{-3}$ and $\Gamma = 3$ are shown in Figure 2.5 where the best solution equates to the sum of lengths over four routes. Increasing the maximum number of Monte Carlo steps results in the optimal solution (i.e. 681) being obtained when $M_C > 1\times10^6$.

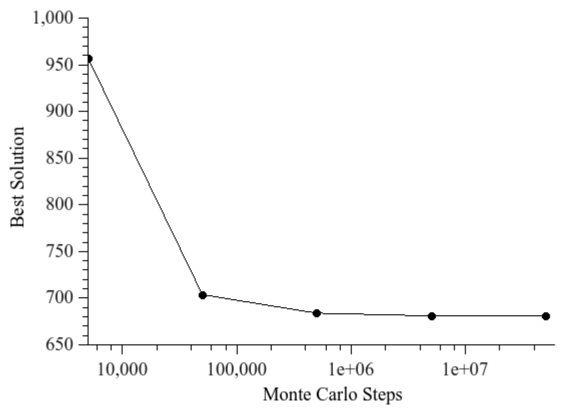


Figure 2.5: The effect of increasing the number of Monte Carlo Steps upon convergence towards the optimal solution.

Looking at the wall clock times for QACVRP for the benchmark instances shown in Table 2.1, observe that in some instances this is larger than the SA wall clock time but in most cases this results in a higher success rate. It is important to note that because QACVRP obtains the BKS values in all cases, this implies that the choice of the spin encoding scheme to compute the kinetic energy term is effective in exploring the energy landscape of all these VRP instances. The artificially induced fluctuations provide an effective solution diversification strategy guiding the search to unexplored regions.

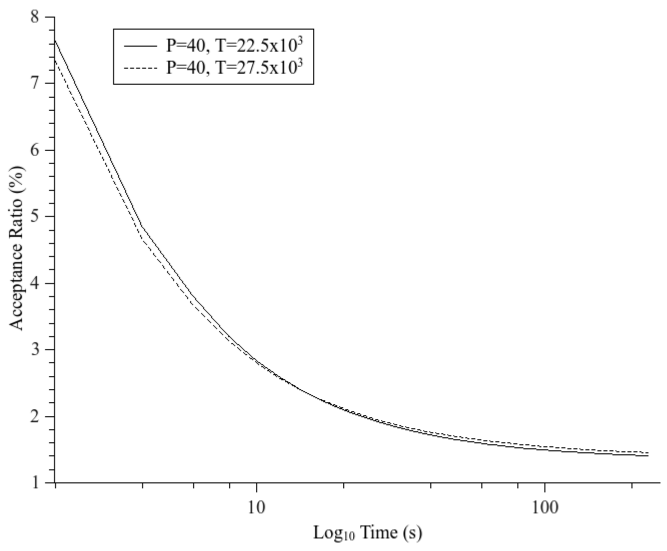


Figure 2.6: Acceptance ratio for two different effective quantum temperatures. The ratio is defined as the number of inferior moves taken, divided by the total number of moves. The lower temperature setting shows a better convergence as there is a higher rejection of inferior moves upon the approach to optimal solution.

Using the parameters $P = 40$, $T = 22.5{\times}10^{-3}$, $\mathit{\Gamma} = 3$ and $M_C = 5{\times}10^6$ the acceptance ratio was calculated for p-n101-k4 by sampling solutions over all iterations. As shown in Figure 2.6 the curve slowly declines over time and then stabilizes to a plateau level. In this case the effective quantum temperature ($PT$) is equal to 0.9 and as seen in Figure 2.4, yields 100% success rate when finding the optimal solution. However, as Figure 2.4 shows, changing the effective quantum temperature to $PT = 1.1$ impacts the solution quality. The same acceptance ratio experiment was repeated with $PT = 1.1$ (i.e. $P = 40$ and $T = 27.5{\times}10^{-3}$) and although the acceptance ratio curve has the same form, a plateau is reached earlier (see Figure 2.6), often leading to premature convergence to a local optimum rather than the global value.

## 2.7 Conclusion

As far as could be determined, this chapter is the first study of the use of quantum annealing for solving the Capacitated Vehicle Routing Problem. An important step in applying QA to a problem is to find an intuitive way to encode the problem in terms of a set of spins, such that operators can be used by the QA algorithm to generate new valid solution configurations. The novelty of this encoding approach is the mapping of VRP connectivity to the spin matrix, and using VRP local neighbourhood operators when attempting to improve a configuration, as this ensures that a valid network is always created. This encoding scheme allows the replicas to communicate between each other in a consistent manner. The performance of the quantum annealing algorithm has been measured by undertaking experiments using standard CVRP benchmarks and compared against a fixed temperature SA algorithm.

The primary conclusion is that the quantum annealing algorithm finds BKS values for all benchmarks used in this study and outperforms SA in terms of success rate in the majority of cases.

# Chapter 3

# Tuning QA for the Vehicle Routing Problem

## 3.1 Introduction

In Chapter 2, Quantum Annealing was applied to the vehicle routing problem and the results were promising. For all benchmark instances, optimal results were obtained. However, 100% success rate was not achieved in every case, and tuning the control parameters for larger instances proved cumbersome. This chapter addresses the remaining difficulties by investigating methods of tuning the parameters of QA using vehicle routing problems. VRP is a class of NP-hard combinatorial optimization problems in which the objective is to find optimal routes (number and length) used by a fleet of vehicles, each with a given capacity and stationed at one or more depots, to serve a set of customers and satisfy all demands. The objective is to minimize the number of routes and/or total route length. Each route starts and ends at a depot and the customers are visited only once by one vehicle. The computational effort required to solve a VRP instance increases exponentially with the problem size, and adding constraints such as time windows and back-hauling further complicates the search. Consequently, heuristic methods are employed to obtain approximate solutions, in a reasonable time, that are of sufficient accuracy for practical purposes.

Proposed here are tuning schemes for QA, which could be automated and applied to any problem instance. Behaviour (cost) models are developed and evaluated to determine their usefulness in uncovering relationships between the fitness landscape and the strategy used by the heuristic to navigate it, revealing good values for the controlling parameters. In these regards, a factorial Design of Experiments (DoE) method was employed to help identify which control parameters contribute the most to the search strategy, and Fitness Clouds were plotted and used as behaviour models. Further, the research suggests schemes which can be generalized and applied to any heuristic.

VRP is chosen for this work as it is a well understood problem - there exists a large body of research and a wide variety of heuristics have been used to solve it. There are also plenty of familiar benchmark instances with deterministically proven optimal solutions. This allows comparisons between heuristics in absolution terms e.g. success rate. VRP also extends into other domains which have industrial applications such as staff rostering, and the Technician Routing and Scheduling Problem (TRSP).

An empirical approach is taken wherein measurements of run-time behaviour are exploited to transform existing good values of control parameters so that they can be used successfully for other problem instances. The methods shown both simplify hand-tuning so that the heuristic performs successfully when applied to larger instances, and establish control parameter values for instances which belong in broadly defined groupings. In addition, new best known solutions for large-scale instances, and initial results for the distance-constrained variant of the vehicle routing problem are presented.

The previous chapter demonstrated the effectiveness of Quantum Annealing (QA) for solving many instances of the Capacitated Vehicle Routing Problem (CVRP). Optimal results were obtained for all benchmark instances by applying a single set of values for the algorithm's control parameters - values which were methodically determined to achieve the maximum success rate for a reference instance. The success rate is the percentage of the number of times the algorithm finds the best known score for a given instance over a number of runs.

Table 3.1 shows an excerpt of the instances for which this parameter set was unable to achieve 100% success rate. An indication of the complexity of each instance can be inferred from the name. P-n101-k4 for example, has 101 nodes/customers served by 4 vehicles whereas M-n121-k7 has 121 served by 7 (the initial letter indicates the benchmark collection from which the instance is drawn). Notably, the scores for smaller instances were much lower than for the reference instance. This is contrary to intuition, that one might expect parameters giving the best results for a larger instance would perform easily as well for smaller instances. (One may expect also that parameters for smaller instances will not work well for larger ones.) Given that the local search method is effective enough to allow the metaheuristic to find the optimal solution in at least 11% of the experiments, and that many of the instances appear less complex than the reference, one can conclude that the values of the control parameters are incorrect. If the temperature value is set too high or the magnetic field is too strong, convergence to a minimum is slowed down or inhibited completely. If set too low, the rate of convergence is high and entrapment at poor local minima is likely. If the population size is too small, the search of solution space covers only a reduced area which may not contain optimal solutions. It seems clear that the universal application of a single set of control parameters will not guarantee consistently good performance and the algorithm requires tuning on a case-by-case basis.

Table 3.1: An excerpt of Table 2.1) showing that the parameter set tuned in Chapter 2 was not globally applicable. QACVRP was not able to deliver 100% success rate (number of optimal solutions found) in about half of the chosen experimental instances.

| **Instance** | **QA Success %** | **SA Success %** |
|---|---|---|
| **P-n101-k4 (reference)** | **100** | **100** |
| P-n50-k10 | 63 | 28 |
| P-n55-k10 | 35 | 31 |
| P-n60-k15 | 79 | 79 |
| P-n70-k10 | 78 | 61 |
| P-n76-k4 | 52 | 44 |
| P-n76-k5 | 87 | 22 |
| B-n63-k10 | 26 | 25 |
| B-n66-k9 | 91 | 44 |
| B-n67-k10 | 42 | 88 |
| B-n68-k9 | 69 | 87 |
| B-n78-k10 | 97 | 99 |
| M-n121-k7 | 90 | 76 |
| F-n135-k7 | 11 | 4 |

How then does one tune metaheuristic control parameters for best results? One could apply the tuning methodology for every instance, providing a specific set of control parameters for each. To save time, the processes of the methodology could be captured, encoded, and then left to a computer program to automatically decide the parameters. These approaches work because feedback can be derived from information known beforehand about the optimal result. Benchmark instances are often supplied with deterministically proven optimal solutions. However, for larger benchmarks, and in dynamic or industrial applications where problem instances are created in real-time, such information is limited or non-existent.

Additional tuning difficulties are presented by metaheuristics with two or more control parameters, each of which may be tightly interdependent. For example, the coupling term used in QA is a non-linear function of magnetic field strength and effective temperature. This term is extremely sensitive to variations in either parameter, and tuning is further complicated because the Metropolis criteria [15] is simultaneously dependent upon temperature. A Design of Experiments (DoE) method [57] can be helpful in uncovering major dependencies between such variables, but a course of factorial experiments can be time-consuming, and predicting the ranges for numerous and sensitive variables is difficult without once again resorting to guesswork or serially hand-tuning.

It is for reasons like these that metaheuristics with fewer control parameters are attractive - they are simpler to tune. Late Acceptance Hill Climbing [37] has a single parameter controlling the size of a fitness array which acts as a 'memory' of good solutions. Cuckoo Search is reported [58] to be superior to Genetic Algorithms in part because of having only two parameters - nest abandonment rate and population size. In QA, it has been shown [24] that the number of parameters can be reduced by one, by setting the magnetic field value to be constant. This idea can be greatly extended by making the whole coupling term a constant, thereby removing the mutual dependence of the effective temperature and magnetic field parameters. With the key parameters uncoupled from one another, time is saved when determining their values by hand. Large-scale problems and instances of the Distance-constrained Capacitated Vehicle Routing Problem (DCVRP) may be tackled without tedium. Furthermore, if some means other than hand-tuning can predict the value of temperature, a single variable remains to be tuned - the replica count (population size).

## 3.2 Quantum Annealing

Quantum Annealing is an energy-based metaheuristic which uses the Path-Integral Monte Carlo (PIMC) method [23] to approximate the ground state of the Ising Model. The fitness function is described (3.1) by the Hamiltonian

$$H = H_p + H_k \tag{3.1}$$

Where the cost $H$ is the sum of potential energy $H_p$ and fluctuations in kinetic energy $H_k$. $H_k$ is the term which represents the quantum mechanical phenomenon of 'tunneling', where a particle trapped in a low energy state, can 'tunnel' through high potential barriers into a lower state. This effect can be simulated in a metaheuristic by using an Ising Model representation of the optimization problem. In simple terms, this is maintaining a population $P$ of simultaneously evolving solutions called replicas, where $H_k$ is calculated from an interaction between adjoining replicas.

When QA is applied to an optimization problem, $H_p$ takes the role of the cost of a solution (for VRP, see (3.4)), while $H_k$ is a scaled sum of the spin interactions between $P$ neighbouring solutions held in a circular list.

$$H_k = J_\Gamma \sum_P \sum_i \sigma_{P-1,i} \sigma_{P,i} \sigma_{P+1,i} \tag{3.2}$$

Each replica represents the solution as a spin matrix $\sigma$ containing $i$ elements which can assume values of $\{-1, +1\}$. The interaction energy between the spins of adjoining replicas is generated by the term, $\sigma_{P-1,i} \sigma_{P,i} \sigma_{P+1,i}$. This is formulated in such a way that it is maximally positive when spins are aligned, tending negatively otherwise. In this respect, the interaction energy can be thought of as a measure of similarity. If the replicas are identical, the energy is higher.

$J_\Gamma$ is the coupling term which is normally varied during the annealing process via adjustments to the magnetic field strength $\mathit{\Gamma}$, amplifying or attenuating the interactions between replicas.

$$J_\Gamma = \frac{-T}{2} \ln \tanh \left( \frac{\mathit{\Gamma}}{PT} \right) \tag{3.3}$$

Consequently, this contributes towards the acceptance of $H$ in the Metropolis criteria.

### 3.2.1 QA for CVRP, and the PT tuning method

CVRP is a variant of VRP in which all vehicles are subject to the same capacity constraint $Q$. CVRP is an undirected graph $G = (V, E)$ consisting of the vertex set $V = \{v_0, v_1, ...v_n\}$ and edge set $E = \{(v_i, v_j)|v_i, v_j \in V, i < j\}$. The restriction $i < j$ ensures the distance between a pair of vertices is identical in both directions. The first vertex is usually considered to be the depot from which a fleet of trucks $m$ serves $n$ customers, whose locations are represented by a vertex set, and have varying demands for goods $q_i$ . The goal is to minimize the number of routes and/or total distance traveled by the trucks $d_{ij}$.

QA for CVRP (QACVRP) uses a two-dimensional spin matrix in which the elements represent customer-customer connections that form routes for each truck. A non-zero cell in the matrix indicates a path between two vertices and because these connections are bi-directional, the spin matrix is symmetric. Figure 2.1 in Chapter 2 shows this arrangement using an example of 15 customers serviced by 4 vehicles. The hexadecimal value shows how rows of connections, each expressed as a single bit, may be encoded into a single memory word. In any one replica, the classical potential energy is the total length of all routes and so $H_p$ for the whole ensemble is given by (3.4).

$$H_p = \sum_{P}\sum_{i,j} d_{P,i,j}\,\sigma_{P,i,j} \tag{3.4}$$

The local search scheme used in QACVRP is comprised of a set of neighbourhood operators $N$=\{*Move, Swap, Move-string, Swap-string, 2-Opt, 2-Opt$*$*\} from which one is selected at random and applied repeatedly until a feasible configuration is found. The Move operator selects a customer and transplants randomly to another position. The Swap operator selects two customers at random and exchanges them. Array-based counterparts of these operators are Move-string and Swap-string which work with randomly sized sets of contiguous customers instead of individuals. The 2-Opt operator selects at random two non-adjacent edges of a single route and then reverses the connection order of the customers between the outer endpoints. 2-Opt$*$ exchanges randomly-sized end portions of two routes, preserving the order of the connections between customers.

Figure 3.1 shows the QACVRP implementation used previously. Lines 1 and 2 initialize the replicas with randomized feasible solutions, setting the current best solution from the first replica. Lines 3 and 24 define the outer loop which terminates when the number of remaining iterations (Monte Carlo steps) reaches zero in line 23. For each iteration of the outer loop, $J_\Gamma$ is calculated from the current parameter values (line 4) and the replica index is reset (line 5).

**Variables**: $P$: number of replicas, $T$: temperature, $\mathit{\Gamma}$: magnetic field, $\Delta\mathit{\Gamma}$: decrement value of magnetic field, $M_C$ : number of Monte Carlo steps, $N$: set of neighbourhood operators, $n$: chosen operator, $S$: set of replicas, $S_{best}$ : best solution, $s'$: candidate solution, $z$: index of current replica.

```
 1: $S \leftarrow$ circular list of feasible randomized solutions
 2: $S_{best} \leftarrow S_0$
 3: **while** $M_C > 0$ **do**
 4:     $J_\Gamma \leftarrow (-T/2)\ln(\tanh(\mathit{\Gamma}/PT))$
 5:     $z \leftarrow 0$
 6:     **while** $z < P$ **do**
 7:         randomly choose $n \in N$
 8:         $s' \leftarrow n(S_z)$
 9:         $\Delta H_p \leftarrow H_p(s') - H_p(S_z)$
10:         $\Delta H_k \leftarrow$ Interact$(S_{z-1}, s', S_{z+1})$ − Interact$(S_{z-1}, S_z, S_{z+1})$
11:         $\Delta H \leftarrow (\Delta H_p/P) + J_\Gamma \Delta H_k$
12:         **if** $((\Delta H_p <= 0)$ **or** $(\Delta H <= 0))$ **then**
13:             $S_z \leftarrow s'$
14:         **else if** $\exp(-\Delta H/T) > \text{random}(0,1)$ **then**
15:             $S_z \leftarrow s'$
16:         **end if**
17:         **if** $H_p(S_z) < H_p(S_{best})$ **then**
18:             $S_{best} \leftarrow S_z$
19:         **end if**
20:         $z \leftarrow z + 1$
21:     **end while**
22:     $\mathit{\Gamma} \leftarrow \mathit{\Gamma} + \Delta\mathit{\Gamma}$
23:     $M_C \leftarrow M_C - 1$
24: **end while**
25: return $S_{best}$
```

Figure 3.1: Quantum Annealing for Capacitated Vehicle Routing Problems.

Lines 6 and 21 define the inner loop which terminates when the replica index reaches the number of replicas in line 20. Line 7 selects an operator at random from the set of neighbourhood operators. On line 8 the solution belonging to the replica which is currently indexed is modified by the chosen operator until a feasible candidate is returned. The difference in potential energy is calculated on line 9 using the candidate and the replica solutions. On line 10 the difference in kinetic energy is calculated by computing the interaction terms (3.2) between the candidate and the replica solutions. The total change in energy is calculated on line 11. Line 12 checks if the difference in potential energy or total energy has decreased and if so, the candidate solution is assigned to the current replica (line 13). If the energy difference has increased (the solution cost is worse) the candidate solution can still be accepted (line 15) if the probabilistic check on line 14 is passed. The acceptance of a worse solution is controlled by the change in total energy and the temperature (Metropolis criteria). On line 17 the cost of the current replica solution is calculated and compared with the best recorded solution. If better, it is assigned as the best solution on line 18.

Once all replicas have been updated in this manner, the inner loop relinquishes control to the outer whereupon the magnetic field value is adjusted (line 22) for use in the next iteration. Once all outer loop iterations are complete, the best solution found is returned on line 25.

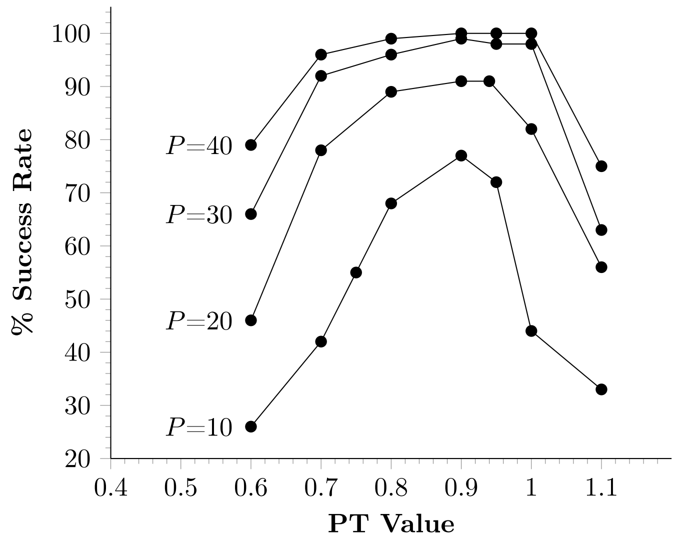


Figure 3.2: $PT$ tuning using instance P-n101-k4. $\Gamma$ and $P$ are held constant while $T$ is increased. The success improves with $P$ until a threshold determined by $T$ is reached. Success rate is defined as the number of optimal solutions delivered in each set of experiments. How should an effective value of $T$ be chosen without resorting to such a series of experiments?

Previously, QACVRP was employed to solve CVRP instances with a single set of control parameters determined using a method entitled PT Tuning (effective quantum temperature tuning)

*PT Tuning.* A factorial DoE study was first undertaken to expose any dominance amongst control parameters for the chosen variable assignments (Table 3.2). Since there are five parameters and two adjustments (+1/-1) allowed to their nominal values, $2^5$ experiments were performed, each consisting of 100 attempts to solve the same CVRP instance. Table 3.3 shows the effect of parameter interactions upon average solution cost. From this, contributions can be calculated for all possible interaction sets. For any chosen set, an average is taken from the scores where the product of the adjustments is equal to 1, and another is taken where the product is equal to -1. The two averages are used as endpoints to form a line, the steepness of which indicates the size of the contribution for the chosen set. The absolute gradient value for each parameter interaction set is presented in Figure 3.3.

Although this shows that $P$ alone makes the greatest single contribution, it is awkward to adjust $P$ and maintain good success without also making compensatory adjustments to $T$ and $\Gamma$. (This observation motivated subsequent research efforts to uncouple the parameters to ease tuning). However, the study supports the idea of tuning $P$ and $T$ as a single term with the evidence that $PT$ makes the second greatest contribution to average solution cost. Monte Carlo steps $M_C$, and then magnetic field start and end values $(G_0, G_1)$ are the next largest contributors. Naturally, larger values of $M_C$ improve results by allowing longer search times. Temperature by itself has the lowest single contribution, with the effects of $T$ likely being swamped by high values assigned to the magnetic field (3.3).

Table 3.2: Variable assignments for factorial experiments. Nominal values specify the baseline setting for each parameter. Each experiment in the factorial study was conducted with baseline offset by the $\{+1, -1\}$ values for each paramter.

| | $T$ | $G_0$ | $G_1$ | $P$ | $M_C$ |
|---|---|---|---|---|---|
| **Nominal** | 80 | 3000 | 1050 | 10 | 5,000,000 |
| **+1** | 2.5 | 500 | 500 | 4 | 500,000 |
| **-1** | -2.5 | -500 | -500 | -4 | -500,000 |

Table 3.3: Results of factorial experiments using CVRP instance P-n101-k4. Each row represents 100 runs of the QACVRP algorithm, with the baseline parameter values adjusted for all combinations of the $+1, -1$ variables (see Table 3.2).

| $T$ | $P$ | $G_0$ | $G_1$ | $M_C$ | Av. Score |
|---|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 | 685.608 |
| -1 | -1 | -1 | -1 | +1 | 684.863 |
| -1 | -1 | -1 | +1 | -1 | 685.585 |
| -1 | -1 | -1 | +1 | +1 | 685.485 |
| -1 | -1 | +1 | -1 | -1 | 685.748 |
| -1 | -1 | +1 | -1 | +1 | 685.580 |
| -1 | -1 | +1 | +1 | -1 | 685.954 |
| -1 | -1 | +1 | +1 | +1 | 684.804 |
| -1 | +1 | -1 | -1 | -1 | 683.009 |
| -1 | +1 | -1 | -1 | +1 | 682.610 |
| -1 | +1 | -1 | +1 | -1 | 682.012 |
| -1 | +1 | -1 | +1 | +1 | 681.913 |
| -1 | +1 | +1 | -1 | -1 | 682.312 |
| -1 | +1 | +1 | -1 | +1 | 681.997 |
| -1 | +1 | +1 | +1 | -1 | 681.758 |
| -1 | +1 | +1 | +1 | +1 | 681.579 |
| +1 | -1 | -1 | -1 | -1 | 685.134 |
| +1 | -1 | -1 | -1 | +1 | 684.793 |
| +1 | -1 | -1 | +1 | -1 | 685.542 |
| +1 | -1 | -1 | +1 | +1 | 685.055 |
| +1 | -1 | +1 | -1 | -1 | 685.140 |
| +1 | -1 | +1 | -1 | +1 | 684.523 |
| +1 | -1 | +1 | +1 | -1 | 684.895 |
| +1 | -1 | +1 | +1 | +1 | 684.945 |
| +1 | +1 | -1 | -1 | -1 | 684.128 |
| +1 | +1 | -1 | -1 | +1 | 683.732 |
| +1 | +1 | -1 | +1 | -1 | 683.084 |
| +1 | +1 | -1 | +1 | +1 | 682.725 |
| +1 | +1 | +1 | -1 | -1 | 682.924 |
| +1 | +1 | +1 | -1 | +1 | 682.609 |
| +1 | +1 | +1 | +1 | -1 | 682.561 |
| +1 | +1 | +1 | +1 | +1 | 682.224 |

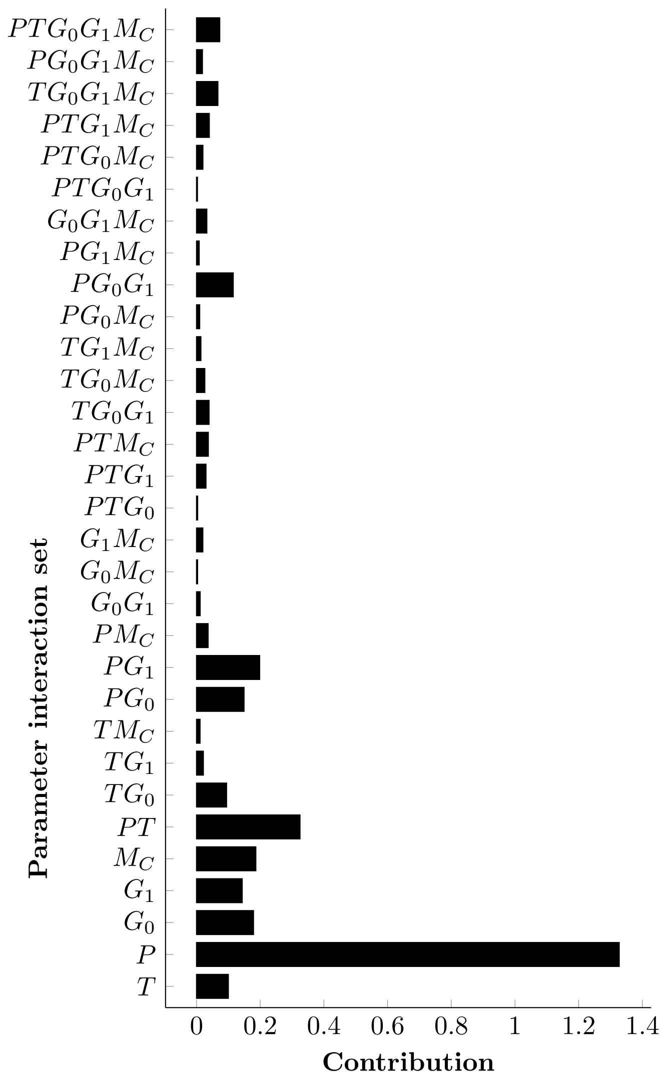


Figure 3.3: Summary of the factorial experiments which show the main and interaction (combined) effects of parameters $P$, $\Gamma(G_0, G_1)$, $T$ and $M_C$ upon solution cost. Adjusting $P$ and $T$ together as single term, while leaving the other paramters constant, is the second-best approach to tuning. Adjusting $P$ alone gives the best, but it is important to somehow determine $T$.

As shown in [24], $\Gamma$ was held constant during $PT$ tuning whilst suitable values for temperature $T$ and $P$ were determined by experimentation. The effective quantum temperature $PT$ was plotted (Figure 3.2) against the success rate over a series of experiments using a reference instance. The reference instance was selected because it has the approximate median number of customers within the chosen benchmarks of [41], which range from 50 to 262 customers. This was deemed to be of average complexity since there are 91 instances with fewer than 100 customers and only

10 instances with 100+ customers [53]. While varying $T$ against several fixed values of $P$, the value of $PT$ which gave the best success rate was noted. The values of $\mathit{\Gamma}$, $P$, and $T$ at this point formed the reference control parameter set $C_{ref} = \{\mathit{\Gamma}_{ref}, P_{ref}, T_{ref}\}$. It was then assumed that applying $C_{ref}$ to any benchmark instance in the study would yield good success rates.

### 3.2.2 Simplifying tuning: Parameters uncoupled

When attempting to tune the algorithm for use with larger or more complex instances, the difficulty of adjusting non-linearly codependent variables is compounded with the need to allow increased run times.

Effective quantum temperature and magnetic field strength are the variables used to calculate the coupling term $J_\Gamma$ which scales the energy generated by interactions of spins amongst the replica ensemble. Consequently, the value of $H_k$ is highly sensitive to small changes in $P$, $T$ or $\mathit{\Gamma}$, and because $T$ is also the variable which governs probabilistic acceptance in the Metropolis criteria, tuning is frustrating. This situation is exacerbated when larger instances are involved. The computational workload increases exponentially with problem size, and so results which can provide feedback to re-tune the parameters are delivered at longer intervals.

If $J_\Gamma$ was established beforehand and kept constant whilst annealing, $\mathit{\Gamma}$ can be ignored while $T$ would have a role limited to governing thermal effects via the Metropolis criteria. In QACVRP, the Hamiltonian took the form

$$H = \frac{\Delta H_p}{P} - J_\Gamma \Delta H_k \tag{3.5}$$

Removing the averaging term $P$ from (3.5) uncouples another control parameter and would further simplify tuning - $P$ could be increased, regardless of temperature, to improve the success rate. The Hamiltonian for such an arrangement is shown in (3.6).

$$H = \Delta H_p - J_\Gamma \Delta H_k \tag{3.6}$$

The isolation of $T$ requires that the annealing process be divided into two phases, each separately taking advantage of $H_p$ and $H_k$. In the first phase, higher temperatures may be chosen so that $H_p$ dominates $H_k$ in (3.6). This 'thermal' phase results in a process akin to

Simulated Annealing (SA) but which includes small, non-negligible quantum fluctuations. In the second phase, the temperature should be drastically lowered so that $H_k$ dominates in $H$. In this 'quantum' phase, optimization relies almost exclusively upon the state of the replica ensemble and the interactions therein. In both phases, $P$ may be adjusted without incurring the need to re-tune the other parameters. It may be increased to improve the accuracy of PIMC, with the expectation that a better result may be found at the expense of run time. Conversely, $P$ may be decreased to speed up annealing at the expense of accuracy.

This two-phase approach may be employed to ease the tuning process, by repeating the thermal phase with small values for $P$ and $M_C$ to establish a good value for $T$. With $T$ established, a final run of the first phase can be done with increased $P$ and $M_C$. The resulting solution from this run may be supplied as a starting point for the quantum phase, in a similar fashion to a construction algorithm which supplies an improvement algorithm with an initial solution.

Figure 3.4 shows the revised QA algorithm which uses a fixed value of $J_\Gamma$ and on line 10, the altered Hamiltonian (3.6).

**Variables**: $P$: number of replicas, $T$: temperature, $J_\Gamma$: magnetic coupling strength, $M_C$ : number of Monte Carlo steps, $N$: set of neighbourhood operators, $n$: chosen operator, $S$: set of replicas, $S_{best}$ : best solution, $s'$: candidate solution, $z$: index of current replica.

```
1: $S \leftarrow$ circular list of feasible randomized solutions
2: $S_{best} \leftarrow S_0$
3: while $M_C > 0$ do
4:    $z \leftarrow 0$
5:    while $z < P$ do
6:       randomly choose $n \in N$
7:       $s' \leftarrow n(S_z)$
8:       $\Delta H_p \leftarrow H_p(s') - H_p(S_z)$
9:       $\Delta H_k \leftarrow \text{Interact}(S_{z-1}, s', S_{z+1}) - \text{Interact}(S_{z-1}, S_z, S_{z+1})$
10:      $\Delta H \leftarrow \Delta H_p + J_\Gamma \Delta H_k$
11:      if $((\Delta H_p <= 0)$ or $(\Delta H <= 0))$ then
12:         $S_z \leftarrow s'$
13:      else if $\exp(-\Delta H/T) > \text{random}(0,1)$ then
14:         $S_z \leftarrow s'$
15:      end if
16:      if $H_p(S_z) < H_p(S_{best})$ then
17:         $S_{best} \leftarrow S_z$
18:      end if
19:      $z \leftarrow z + 1$
20:   end while
21:   $M_C \leftarrow M_C - 1$
22: end while
23: return $S_{best}$
```

Figure 3.4: Fixed $J_\Gamma$ Quantum Annealing for Capacitated Vehicle Routing Problems.

### 3.2.3 Determining temperature: Energy-based Scaled Parameter Tuning

With the parameters uncoupled from one another in FJ-QACVRP (*Fixed $J_\Gamma$ Quantum Annealing for the Capacitated Vehicle Routing Problem*), a systematic means of establishing $T$ for the first annealing phase would be of benefit when tackling groups of problem instances, such as the benchmarks of Augerat et al [59] as attempted by QACVRP under the *PT* Tuning scheme. It would be ideal to identify and then reproduce for any instance, the runtime behaviour of the algorithm while it is successfully (and consistently) solving a reference instance.

Even though success rates in Table 3.1 were inconsistent, there is promise in using $C_{ref}$ since optimal results were found for every instance. To improve this, some factor needs to be discovered which relates a subject to a reference instance. This factor would have to reliably transform $C_{ref}$ into a subject parameter set $C_{subj} = \{\mathit{\Gamma}_{subj}, P_{subj}, T_{subj}\}$. To find this factor, the fitness landscape [6] of the problem needs to be considered.

The fitness landscape for a problem instance is the abstract topology formed from the combination of all possible solutions, an objective function, and neighbourhood operators. For a combinatorial optimization problem like CVRP, it is impossible to visualize this landscape. Each point in the landscape represents a possible solution and is a vector composed of the solution cost and configuration, and must be connected to all possible neighbouring solutions. As a whole, this multidimensional topology is difficult to conceive and so must be greatly simplified if to be of any practical help in discovering relationships between instances. To assist with this discovery, the fitness landscape is reduced to an energy landscape. Measurements of potential energy $H_p$ can be recorded and these are equivalent to the cost component of vertices in the landscape. Kinetic energy $H_k$ can also be recorded, but is of less immediate use as a measure of similarity, it gives indirect evidence of how connections between vertices are grouped in the landscape. (In a single phase of FJ-QACVRP with high $T$, Hamiltonian energy $H$ is imagined to form a similar topology to that formed by $H_p$, given that $H_k$ makes a relatively tiny contribution to the total energy.) The energy landscape which is explored by a metaheuristic while it solves a problem can be visualized in the form of a scatterplot or Fitness Cloud [60]. The geometric features of the plot give a visual indication of the run-time behaviour of the algorithm, and therefore may be regarded as a dynamic cost model [6]. Fitness Clouds were originally invented to study how mutation and crossover operators caused solutions to evolve in Genetic Programming. They were further analysed to produce metrics for characterizing and measuring the difficulty of optimization problems [61]. For this work, they provide visual substance to support the proposal that there are commonalities in the energy landscapes of different problem instances which may be exploited to assist with tuning.

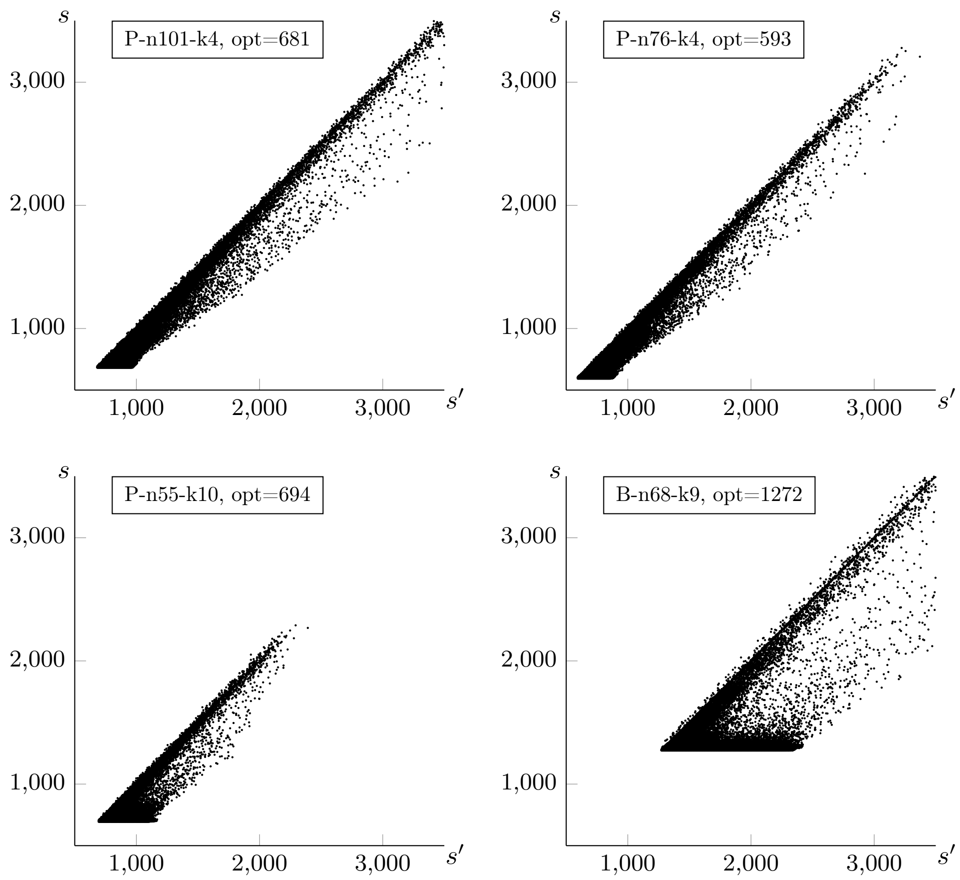


Figure 3.5: Fitness clouds of instances sampled from the benchmarks of Augerat et al [59]. They were generated by QACVRP and are similar in structure. Each data point represents the change in cost (energy) as the optimizer steps from the current solution $s$ to the next $s'$. When the optimal cost *opt* was reached or when the number of Monte Carlo iterations were exhausted, the sampling of the QACVRP landscape was halted.

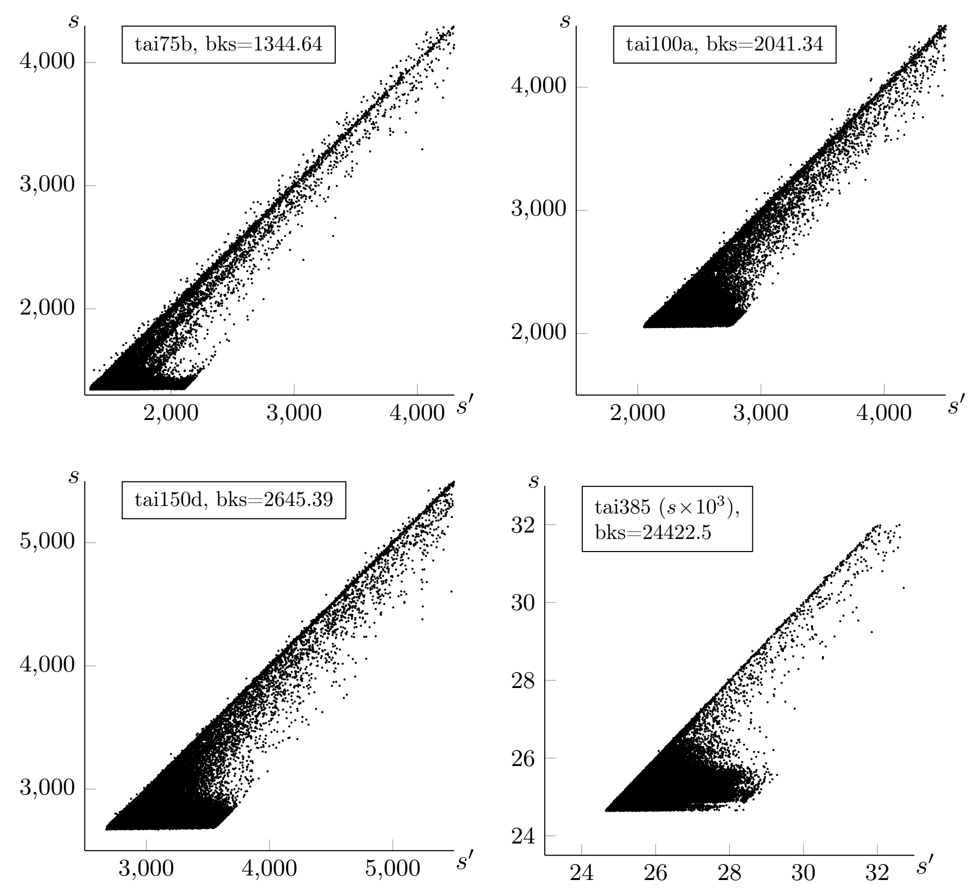


Figure 3.6: Fitness clouds sampled from the instances in the benchmarks of Taillard [62]. Similar as with the previous Figure 3.5, they were generated by QACVRP the sampling of which was halted when the best-known score *bks* was attained or exceeded, or when the Monte Carlo simulation ended.

Several plots were made from the recordings of energy values as QACVRP solved selected CVRP instances. A cursory analysis of the plots show that in each case, there is quick convergence to, and an intensive search of a region around the optimal solution. More encouragingly, there is a good similarity in shape and structure, which lends support to the idea that one landscape may be approximately transformed into another through the use of scaling or affine transformations. (In CVRP, the fitness cloud shows a distribution of the changes to the geometry of the routes, with $\Delta E_p$ essentially being a change in distance. Therefore, it should be possible to remap one instance to another.) With such transformations, the aim is to adapt a set of parameters for a reference instance for use in a subject.

For the first phase of FJ-QACVRP, we propose that a useful factor which correlates fitness landscapes is some measure or function of $\Delta H_p$ since it has the greatest impact (3.1) in the Metropolis criteria. If the ratio of such measures from subject and reference instances were known, and since the coupling term $J_\Gamma$ (3.3) is now constant, then $T$ for the subject instance can be calculated as the product of this ratio and the reference effective temperature $P_{ref}T_{ref}$. This would then remove the necessity of tuning the temperature parameter. Peak changes in accepted potential energy $\Delta H_p$ have already been sampled for several CVRP instances using QACVRP.

The coupling term $J_\Gamma$ can be thought of as a factor which amplifies or attenuates similarities between population members (3.2). A good value for $J_\Gamma$ was discovered in the course of determining the parameters of the reference instance, P-n101-k4 in [41]. The value of $J_\Gamma$ which was successful for solving one instance may be used as a constant in solving another, if the temperature is suitably scaled.

When a subject instance is optimized by FJ-QACVRP, the peak change in accepted potential energy $\Delta E_{subj}$ is scaled by the normalization constant (3.7) to provide the value of temperature $T_{subj}$ (3.8).

$$k = \frac{P_{ref}T_{ref}}{\Delta E_{ref}} \tag{3.7}$$

$$T_{subj} = k\Delta E_{subj} \tag{3.8}$$

The following is a brief, stepwise description of the new tuning method, tentatively entitled *Energy-based Scaled Parameter Tuning* (ESPT):

1. With QACVRP solving the reference instance, use the *PT* tuning method (or other) to identify the parameter set $C_{ref}$ which gives the best possible success rate.

2. With QACVRP solving the reference instance using $C_{ref}$, record the peak change in accepted potential energy $\Delta E_{ref}$. The constant $k$ (3.7) can now be calculated.

3. With QACVRP solving the subject instance using $C_{ref}$, record the peak change in accepted potential energy $\Delta E_{subj}$.

4. Use FJ-QACVRP to solve the subject instance using a constant value of $J_\Gamma$ calculated with

$C_{ref}$, $T_{subj} = k\Delta E_{subj}$, $P_{subj} = P_{ref} + p$ where $p$ is a discretionary increase in the number of replicas.

Simple modifications were made to the acceptance function of the software in order to sample the changes in accepted potential energy. QACVRP then solved selected instances 20 times each with $C_{ref} = \{3, 40, 22.5\times10^{-3}\}$ determined in [41], and with Monte Carlo steps $M_C = 10\times10^6$. Table 3.4 shows measurements made using steps 2 and 3 for several instances listed in Table 3.1.

Table 3.4: Peak changes in accepted potential energy. Changes in cost are captured in the acceptance function of QACVRP while an instance is optimized. The algorithm uses a set of reference parameters - values which produce optimal results 100% of the time for the instance P-n101-k4. The largest recorded change is used to predict the value of temperature parameter in the ESPT method.

| **QACVRP** | |
|---|---|
| **Instance** | $\Delta E$ |
| P-n101-k4(reference) | 13 |
| P-n50-k10 | 15 |
| P-n55-k10 | 14 |
| P-n60-k15 | 14 |
| P-n76-k4 | 17 |
| P-n63-k10 | 16 |
| P-n67-k10 | 15 |
| B-n68-k9 | 17 |

## 3.3 Experimental results

Although multicore processor architectures allow for programs to be written to achieve parallelization of their workload, FJ-QACVRP was designed to take advantage in a different way. Rather than breaking the algorithm into parallelizable components, FJ-QACVRP was coded to place a whole experimental run on a separate thread so that it would be allocated to a single CPU core. This design allows many runs to occur simultaneously and so will output results much more quickly. FJ-QACVRP was coded in C++ using the Qt cross-platform application framework. The processor employed was the Intel Xeon E5-2683 v3, having 14 cores with a clock speed of 2GHz, running the Linux operating system.

### 3.3.1 Large and very large scale results

For each of the moderately sized to very large-scale instances, a batch of (14) runs equal in size to the number of CPU cores was performed simultaneously across a range of temperatures. For speed whilst tuning in this first phase, the number of replicas and Monte Carlo steps were kept low, with $P \leq 20$ and $M_C \simeq 20{\times}10^6$. When the batch was completed, the value of $T$ from the run with the best score was considered the most useful and, if necessary, reused with higher values of $P$ to seek further improvements.

The best solution from phase 1 was used to initialize each replica in phase 2. A percentage of the replica ensemble was perturbed by applying a randomized selection of the neighbourhood operators for a fixed number of iterations. A low temperature $T = 0.14$ and higher values of $P$ (40, 60, 80, 160) and $M_C = 50{\times}10^6$ formed the parameter set for runs in the second phase of FJ-QACVRP.

The results of this method are shown in tables 3.5 to 3.9 and any improvements to the best known scores in literature are shown in bold. Combined run times are shown together with the total number of iterations $M_C$ taken for each instance.

Table 3.5: Computational results of FJ-QACVRP using CVRP instances of Taillard [62].

| | | | Phase 1 | | Phase 2[a] | | Total | |
|---|---|---|---|---|---|---|---|---|
| Problem | n | Best | Score | Time (s) | Score | Time (s) | Time | $M_C\times10^6$ |
| tai75a | 75 | 1618.36[b] | 1618.357 | 7 | - | - | 0:00:07 | 0.031 |
| tai75b | 75 | 1344.64[b] | **1344.619** | 4 | - | - | 0:00:04 | 0.019 |
| tai75c | 75 | 1291.01[b] | 1291.008 | 5 | - | - | 0:00:05 | 0.033 |
| tai75d | 75 | 1365.42[b] | 1365.419 | 6 | - | - | 0:00:06 | 0.021 |
| tai100a | 100 | 2041.336[b] | 2043.405 | 92 | 2041.337 | 3 | 0:01:35 | 0.213 |
| tai100b | 100 | 1939.9[63] | 1939.904 | 1144 | - | - | 0:19:04 | 2.778 |
| tai100c | 100 | 1406.202[b] | 1406.965 | 721 | 1406.202 | 2 | 0:12:03 | 1.487 |
| tai100d | 100 | 1581.25[b] | 1582.112 | 156 | **1580.458** | 269 | 0:07:05 | 1.183 |
| tai150a | 150 | 3055.23[b] | 3057.395 | 608 | 3055.232 | 366 | 0:16:14 | 2.538 |
| tai150b | 150 | 2727.67[63] | 2730.304 | 958 | 2727.669 | 24618 | 7:06:16 | 14.478 |
| tai150c | 150 | 2341.84[b] | 2361.424 | 1754 | 2358.659 | 1989 | 1:02:23 | 12.898 |
| tai150d | 150 | 2645.39[b] | 2647.713 | 28146 | 2645.391 | 1 | 7:49:07 | 37.942 |
| tai385 | 365 | 24422.5[64] | 24449.046 | 235760 | **24395.411** | 84561 | 88:58:41 | 138.535 |

[a] 50% replicas perturbed for 5 iterations before annealing.
[b] As reported by Alba et al. [65].

Table 3.5 shows the results of FJ-QACVRP on the CVRP instances in the benchmarks of Taillard [62]. In all but one instance (tai150c) the best known score was found or improved upon (tai75b and tai385), and the majority required both optimization phases. In phase 1, temperature ranges were bounded by $1.18 \leq T \leq 1.82$ with intervals of 0.04. Combined run times ranged from 7 seconds to 65 hours.

Table 3.6: Computational results of FJ-QACVRP using various CVRP instances.

| | | | Phase 1 | | Phase 2[a] | | Total | |
|---|---|---|---|---|---|---|---|---|
| Problem | n | Best | Score | Time (s) | Score | Time (s) | Time (s) | $M_C \times 10^6$ |
| E-n101-k14 | 100 | 1067[64] | 1067 | 216 | - | - | 0:03:36 | 1.564 |
| M-n151-k12 | 150 | 1015[64] | 1015 | 456 | - | - | 0:07:36 | 1.617 |
| M-n200-k16 | 199 | 1274[66] | 1274 | 11709 | - | - | 3:15:09 | 14.004 |
| M-n200-k17 | 199 | 1275[66] | 1275 | 29959 | - | - | 8:19:19 | 13.893 |
| G-n262-k25 | 261 | 5530[67] | 5530 | 78974 | **5526** | 20 | 21:56:34 | 46.060 |

[a] 50% replicas perturbed for 5 iterations before annealing.

Table 3.6 shows the results of FJ-QACVRP on the moderately sized CVRP instances in the benchmarks of Christofides and Eilon [68], Gillett and Johnson [69], and Christofides, Mingozzi and Toth [70]. In phase 1, temperature ranges were bounded by $0.6 \leq T \leq 1.4$ with intervals of 0.05. For all instances except G-n262-k35, the best known score is suspected to be optimal. Since no better results were achieved by significantly increasing the running time or population size of FJ-QACVRP ($M_C \geq 350 \times 10^6$, $P \geq 640$), and because no improvements have been published for over two years, the second phases were omitted. Combined run times ranged from 3.6 minutes to 21.9 hours.

Table 3.7: Computational results of FJ-QACVRP using Golden et al [71] CVRP instances.

| | | | Phase 1 | | Phase 2[a] | | Total | |
|---|---|---|---|---|---|---|---|---|
| Problem | n | Best | Score | Time (s) | Score | Time (s) | Time | $M_C \times 10^6$ |
| 9 | 255 | 583.39[b] | **579.713** | 16738 | - | - | 4:38:58 | 99.984 |
| 10 | 323 | 741.7[64] | **737.512** | 133303 | **737.412** | 71 | 37:02:54 | 42.440 |
| 11 | 399 | 918.45[b] | **914.341** | 84969 | **912.770** | 141287 | 62:50:56 | 23.084 |
| 12 | 483 | 1107.19[b] | **1102.542** | 469021 | **1102.120** | 52312 | 144:48:53 | 88.583 |
| 13 | 252 | 859.11[b] | **857.189** | 19232 | - | - | 5:20:32 | 18.273 |
| 14 | 320 | 1081.31[b] | **1080.830** | 106996 | **1080.553** | 22 | 29:43:38 | 33.608 |
| 15 | 396 | 1345.23[b] | **1340.241** | 140262 | - | - | 38:57:42 | 38.508 |
| 16 | 480 | 1622.69[b] | **1611.503** | 153502 | - | - | 42:38:22 | 68.110 |
| 17 | 240 | 707.79[b] | 707.992 | 39557 | **707.756** | 192 | 11:02:29 | 24.406 |
| 18 | 300 | 998.73[b] | **998.354** | 67251 | **995.984** | 110 | 18:42:41 | 18.509 |
| 19 | 360 | 1366.8578[64] | 1368.291 | 61744 | **1366.591** | 1187 | 17:28:51 | 189.301 |
| 20 | 420 | 1821.15[b] | 1825.057 | 75826 | **1818.949** | 23950 | 27:42:56 | 140.475 |

[a] 50% replicas perturbed for 5 iterations before annealing.
[b] As reported by Alba et al. [65].

Table 3.8: Computational results of FJ-QACVRP using Golden et al [71] DCVRP instances.

| | | | Phase 1 | | Phase 2[a] | | Total | |
|---|---|---|---|---|---|---|---|---|
| Problem | n | Best | Score | Time (s) | Score | Time (s) | Time | $M_C \times 10^6$ |
| 1 | 240 | 5627.54[b] | **5624.119** | 39441 | **5623.469** | 43 | 10:58:04 | 33.042 |
| 2 | 320 | 8447.92[72] | 8464.495 | 12007 | 8447.920 | 4599 | 4:36:46 | 11.112 |
| 3 | 400 | 11036.23[72] | 11047.007 | 56584 | **11036.223** | 8085 | 17:57:49 | 110.986 |
| 4 | 480 | 13624.53[72] | 13632.913 | 51109 | 13624.526 | 26960 | 21:41:09 | 41.205 |
| 5 | 200 | 6460.98[72] | 6460.980 | 39913 | **-** | - | 11:05:13 | 24.670 |
| 6 | 280 | 8412.88[72] | 8412.902 | 2966 | **-** | - | 0:49:26 | 1.143 |
| 7 | 360 | 10195.56[72] | 10200.543 | 5148 | 10195.587 | 2845 | 2:13:13 | 7.849 |
| 8 | 440 | 11663.55[b] | 11681.035 | 29603 | 11672.111 | 70097 | 27:41:40 | 50.312 |

[a] 95% replicas perturbed for 30 iterations before annealing.
[b] As reported by Alba et al. [65].

Tables 3.7 and 3.8 show the results of FJ-QACVRP on the benchmark of Golden et al [71], which include larger sized CVRP and DCVRP instances. For the CVRP instances the temperature ranges for phase 1 were bounded by $0.26 \leq T \leq 1.4$ with intervals of 0.02. FJ-QACVRP performed well and delivered improvements over the best known score in all cases. For problems 9, 13, 15 and 16, scores from the second phase yielded no improvements over the first and are not shown. Combined run times for problems using both phases ranged from 11.2 to 144.8 hours.

For the DCVRP instances, FJ-QACVRP matched the best known in 4 problems and approached to within a negligible fraction of a percent in 3 while new best scores were found for problems 1 and 3. For phase 1 the temperature ranges were within $1.8 \leq T \leq 3$ with intervals of 0.1. Combined run times for problems which required both phases ranged from 4.6 to 27.7 hours.

Table 3.9: Computational results of FJ-QACVRP using Li et al [73] DCVRP instances.

| | | | Phase 1 | | Phase 2[a] | | Total | |
|---|---|---|---|---|---|---|---|---|
| Problem | n | Best[73] | Score | Time (s) | Score | Time (s) | Time | $M_C \times 10^6$ |
| 21 | 560 | 16212.83 | 16232.347 | 12550 | 16212.826 | 59302 | 19:57:32 | 8.303 |
| 22 | 600 | 14641.64 | **14632.952** | 230304 | **14586.109** | 363701 | 165:00:05 | 26.302 |
| 23 | 640 | 18801.13 | 18824.837 | 37464 | 18801.131 | 83571 | 33:37:15 | 17.021 |
| 24 | 720 | 21389.43 | 21411.744 | 78209 | 21389.432 | 12344 | 25:09:13 | 23.895 |
| 25 | 760 | 17053.26 | **16905.233** | 57344 | **16851.976** | 10479 | 18:50:23 | 16.214 |
| 26 | 800 | 23977.74 | 23998.635 | 56315 | **23977.733** | 16977 | 20:21:32 | 43.214 |
| 27 | 840 | 17651.6 | **17621.849** | 266076 | **17508.388** | 51211 | 88:08:07 | 41.357 |
| 28 | 880 | 26566.04 | 26586.149 | 16027 | 26566.035 | 69905 | 23:52:12 | 3.474 |
| 29 | 960 | 29154.34 | 29176.634 | 54894 | 29154.337 | 93270 | 41:09:24 | 7.919 |
| 30 | 1040 | 31742.64 | 31774.909 | 100519 | 31742.640 | 580037 | 189:02:36 | 17.655 |
| 31 | 1120 | 34330.94 | 34361.727 | 287648 | 34330.941 | 239826 | 146:31:14 | 29.521 |
| 32 | 1200 | 36919.24 | 37352.460 | 304073 | 37331.111 | 2139 | 85:03:32 | 25.061 |

[a] 95% replicas perturbed for 30 iterations before annealing.

Table 3.9 show the results of FJ-QACVRP on the benchmark of Li et al which is comprised of procedurally-generated [73] very large-scale DCVRP instances. In all cases but problem 32, FJ-QACVRP equals or improves best known scores. In phase 1 the temperature ranges were within $1.8 \leq T \leq 3.2$ with intervals of 0.1. All attempts required both phases to execute and their combined run times ranged from 18.8 to 189 hours.

### 3.3.2 ESPT results

Table 3.10 shows the outcome of performing step 4 of the ESPT method, in which the temperature value for each instance was predicted for use in FJ-QACVRP. For a valid comparison with QACVRP, the chosen benchmarks were those as used in [41] - sets B and P benchmarks of Augerat et al [59]. Both sets contain instances which cluster the locations of customers and are likely to have similar geometric structures in their fitness clouds.

Using $C_{ref}$ and $\Delta E_{ref}$, $k$ (3.7) and $J_\Gamma$ (3.3) were calculated. For each subject instance, $T_{subj}$ (3.8) was calculated using $k$ and $\Delta E_{subj}$ (Table 3.4). With the parameter set $\{J_\Gamma, P_{ref}+p, T_{subj}\}$ and $p \in [-30, +120]$, FJ-QACVRP was used to solve each subject instance 100 times, after which the success rate was recorded.

In almost all cases, at $P \geq 20$, the success rate is increased beyond that achieved by QACVRP. At $P = 40$ there are significant increases for P-n76-k4, P-n76-k5, P-n55-k10 and B-n63-k10, instances that proved troublesome for QACVRP. Figure 3.7 shows the effectiveness of FJ-QACVRP upon the latter three and with values of $T$ determined beforehand, that $P$ may alone be adjusted until a desired rate/time balance is obtained. Also encouraging for the ESPT method is that for $P \leq 20$ the results almost entirely improve over QACVRP, indicating the potential to improve running speed.

As expected, in all cases increasing $P$ improves the success rate. The temperature component has been scaled using potential energy ratios and successfully applied to subject instances.

Table 3.10: Success rates after application of ESPT.

| | Success % | | | | |
|---|---|---|---|---|---|
| | QACVRP[41] | FJ-QACVRP | | | |
| | P=40 (time) | P=10 | P=20 | P=30 | P=40 (time) |
| P-n101-k4 | 100 (07:56:20) | 99 | 100 | 100 | 100 (00:14:24) |
| P-n40-k5 | 100 (00:00:50) | 100 | 100 | 100 | 100 (00:00:04) |
| P-n45-k5 | 100 (00:01:53) | 100 | 100 | 100 | 100 (00:00:05) |
| P-n50-k7 | 100 (00:09:14) | 100 | 100 | 100 | 100 (00:00:17) |
| P-n50-k10 | 63 (08:27:14) | 62 | 80 | 89 | 99 (00:26:40) |
| P-n51-k10 | 100 (01:12:35) | 91 | 99 | 100 | 100 (00:04:15) |
| P-n55-k7 | 100 (01:11:40) | 100 | 100 | 100 | 100 (00:02:42) |
| P-n55-k10[1] | 35 (10:14:24) | 55 | 87 | 93 | 97 (00:39:21) |
| P-n60-k10 | 100 (01:07:38) | 100 | 100 | 100 | 100 (00:02:42) |
| P-n60-k15 | 79 (08:48:46) | 86 | 97 | 98 | 100 (00:10:06) |
| P-n65-k10 | 100 (01:01:47) | 100 | 100 | 100 | 100 (00:01:12) |
| P-n70-k10 | 78 (13:23:50) | 90 | 100 | 100 | 100 (00:13:38) |
| P-n76-k4 | 52 (14:50:56) | 91 | 100 | 100 | 100 (00:37:15) |
| P-n76-k5[1] | 87 (10:16:25) | 72 | 93 | 98 | 98 (00:57:48) |
| B-n50-k8 | 100 (01:34:54) | 100 | 100 | 100 | 100 (00:03:34) |
| B-n52-k7 | 100 (00:09:06) | 100 | 100 | 100 | 100 (00:00:18) |
| B-n56-k7 | 100 (00:20:34) | 100 | 100 | 100 | 100 (00:01:26) |
| B-n57-k9 | 100 (00:40:50) | 100 | 100 | 100 | 100 (00:01:06) |
| B-n63-k10[1] | 26 (12:49:30) | 25 | 36 | 68 | 68 (01:12:18) |
| B-n64-k9 | 100 (01:04:16) | 100 | 100 | 100 | 100 (00:02:12) |
| B-n66-k9 | 91 (08:56:25) | 96 | 100 | 100 | 100 (00:15:54) |
| B-n67-k10 | 42 (15:42:00) | 100 | 100 | 100 | 100 (00:17:46) |
| B-n68-k9 | 69 (11:30:52) | 93 | 99 | 100 | 100 (00:33:44) |
| B-n78-k10 | 97 (07:10:29) | 100 | 100 | 100 | 100 (00:09:42) |

[1] $P$ value for 100% success shown in figure 3.7.

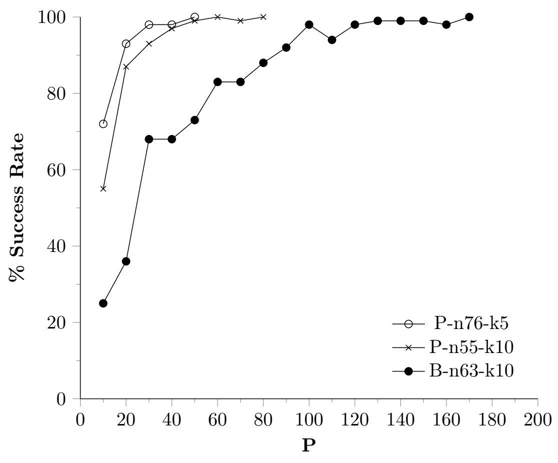


Figure 3.7: With $T$ determined using the ESPT method and $\Gamma$ held constant, the success rate increases in proportion to the last varying parameter, $P$ which may be adjusted to satisfy time/resource constraints. The instances are taken from the benchmark of Augerat et al [59]. Even though B-n63-k10 appears to be of modest complexity, specifying k=10 vehicles to service only n=63 customers, the number of replicas must be set high for maximum success rate to occur. This shows it is a difficult problem instance to solve.

## 3.4 Conclusions

To our knowledge, this section is the first study which reduces the number of control parameters in QA to one (replica count) and which systematically establishes a constant value for the temperature through the use of scaling factors determined from the analysis of fitness clouds.

In principle, the methods and techniques presented could be extended to other vehicle routing problems such as those which use a heterogeneous fleet. A heterogeneous fleet can be represented in a spin matrix by increasing the dimensions by the number of differing vehicles. The additional cells would indicate which vehicle is assigned to each route. Further, the solution representation need not prevent a vehicle going back to the depot to collect more items.

It has been shown that with suitable adjustments to the Hamiltonian and by treating the term which scales interaction energy as a constant, QA is able to tackle very large VRP instances without incurring the need to fine tune all the control parameters. With the tuning much simplified, QA can be used to deliver results which are improvements over, or equal to the best-known scores in the greater majority of large instances of CVRP and DCVRP. The primary conclusion is that the modified QA heuristic is a good match for these kinds of vehicle routing problems.

A reduced number of tunable parameters in FJ-QACVRP presented the opportunity to focus subsequent tuning efforts upon establishing the temperature value through use of the ESPT method. This significantly improves the reliability (in terms of success rate) of QA when dealing with collections of instances which exhibit similar features in their fitness landscapes. For convenience, this study reused existing benchmarks containing instances with similar distributions of customer locations. Naturally, if this method was to be generalized, extending to arbitrary collections of instances, then some effort would need to be spent analysing their fitness clouds in order to group them sensibly prior to the application of ESPT. This would perhaps involve the use of more complicated affine transformations (e.g. skews, mirrors, rotations) rather than scaling factors.

Chapter 4

# Quantum Annealing Algorithm: Enhancements and Variations

## 4.1 Introduction

The Quantum Annealing algorithm was shown in sections 2.5 and 3.3 to be an effective metaheuristic for solving vehicle routing problems, capable of locating the optimal solution when solving most cases of well-known benchmark instances, and competitive with established metaheuristics in terms of solution quality. However, the Quantum Annealing algorithm fares less well when running times are considered. Table 2.1 in Chapter 2 shows that Simulated Annealing achieves an almost-complete dominance over QA (in 22 of 26 instances) when time to find the optimal solution is alone considered. The reason for this is an accurate path-integral calculation requires a large number of solutions, or replicas, to be maintained simultaneously and if this is deployed upon a single thread of execution performance is necessarily degraded.

The key contribution of this chapter is an improved design which, whilst preserving the demonstrated advantages the QA algorithm, divides the processing of the replicas amongst different threads of execution with the goal of reducing the running time by a factor of the number of threads used. This parallelized algorithm, entitled Parallel Quantum Annealing for Vehicle Routing Problems (PQAVRP), is used to solve the same problem instances in the benchmark of Augerat el al [59] as the earlier single-threaded version, resulting in the same solution quality but which is delivered consistently in a fraction of the original running times.

The work in Chapters 2 and 3 proved that the Quantum Annealing (QA) algorithm, in the form of a single-threaded program, was effective for solving the graph colouring problem [24] and the Capacitated Vehicle Routing Problem (CVRP) [41] and its distance-constrained variant (DCVRP) [42]. Through the use of several tuning methodologies which acquired good values for the controlling parameters of the algorthm, optimal results were obtained consistently, and many new best known solutions were discovered. In an academic sense, these results were able to show QA is a viable alternative to other well-known metaheuristics.

One of the strengths of QA is that it is a population-based approach to combinatorial optimization. Through the use of a simulated quantum mechanical tunneling effect, a set of continuously varying solutions, or *replicas*, interact allowing a broader inspection of the solution space. By increasing the number of replica-upon-replica interactions, QA has an increased chance to escape from local minima and so improve the likelihood that good solutions are returned.

However, this strength is also one of the weaknesses of QA. The running time of the algorithm increases in approximate proportion to the number of replicas used, and when larger problems are tackled, can become excessive. For example, it takes over 6 days for the FJ-QACVRP algorithm [42]to solve problems 30 and 31 in the benchmark of Li et al [73] which have over 1000 customers. In such cases, it is important to adjust the number of replicas in order to balance the requirments of solution quality and problem size against an acceptable running time, but this imposes a limit upon the wider practicality of QA.

There are several ways to address the issue of running time. Straight-forward code optimizations (such as unwinding loops, active cache control/awareness and machine-level programming) are some such methods but opportunities for these enhancements are rapidly exhausted whilst being subject to diminishing returns relative to the effort of implementation. Another approach may be to hybridize the algorithm with another metaheuristic which offers improved performance [74][75] in specific program components such as the acceptance function, or local search routines. However, any extra tunable variables from the donor metaheuristic complicate configuration of the hybrid algorithm whilst likely making obsolete the previous tuning schemes developed specifically for QA. A lengthy study, which is beyond the scope of this thesis, would be required to investigate these effects and to also prove a successful hybridization. It is also difficult to adapt any of these methods to take advantage of multi-threaded environments. They usually consist of only a few, tightly-coupled components: a local search scheme (a set of neighbourhood operators); a metaheurisic which guides the local search; and associated support systems (e.g. objective function, I/O). It is not obvious which components are amenable to such fine-grained parallelization or how gains would be achieved in a multi-core environment whilst maintaining cohesive operation. In constrast, population-based methods of optimization are somewhat freed from these constraints, and they offer the opportunity of coarse-grained parallelization at the level of the metaheuristic.

The QA metaheurisic is naturally suited to parallelization. Depending upon the particular implementation of QA, a degree of independence is afforded to each replica. In the simplest case, it is conceivable to provide a dedicated thread of execution for the maintenance of each replica. Further, if such a scheme was designed and engineered sensibly, it would be a solid milestone along the way to distributing the processes for execution in a clustered environment, taking advantage of modern scalable architectures. Parallelization of QA using threading techniques is a sensible initial approach to improving running times, which in principle allows performance gains scaled by the availability of computational resources.

The following sections present a means to significantly reduce the running time of the QA algorithm, removing a major impediment in practical applications. Section 4.2 shows how the algorithm can be reformulated to take advantage of multi-core environments by distributing the workload of the metaheuristic using many threads to host the processing of the population of replicas. The psuedo-code is shown and is implemented for use in experiments in section 4.3, the results of which demonstrate the effectiveness of the multithreaded approach over the previous QA implementations.

## 4.2 Parallelizing the QA Algorithm

The target platform for a parallelized QA algorithm is one which has a CPU with a multicore architecture, and an operating system which provides multithreading support (see appendix A.2, a 14 core processor which hosts Linux). This kind of platform allows the population of replicas to be divided in groups which may be simultaneously processed by individual threads of execution. Communication between replicas must be maintained in order to calculate their interactions.

The calculation of the interaction energy requires that replicas transmit their state, and for this to occur across execution boundaries, a predetermined communication protocol must be established. Such a protocol may be implemented via a mutual exclusion (lock-unlock semantics) mechanism, or a messaging system (semaphores).

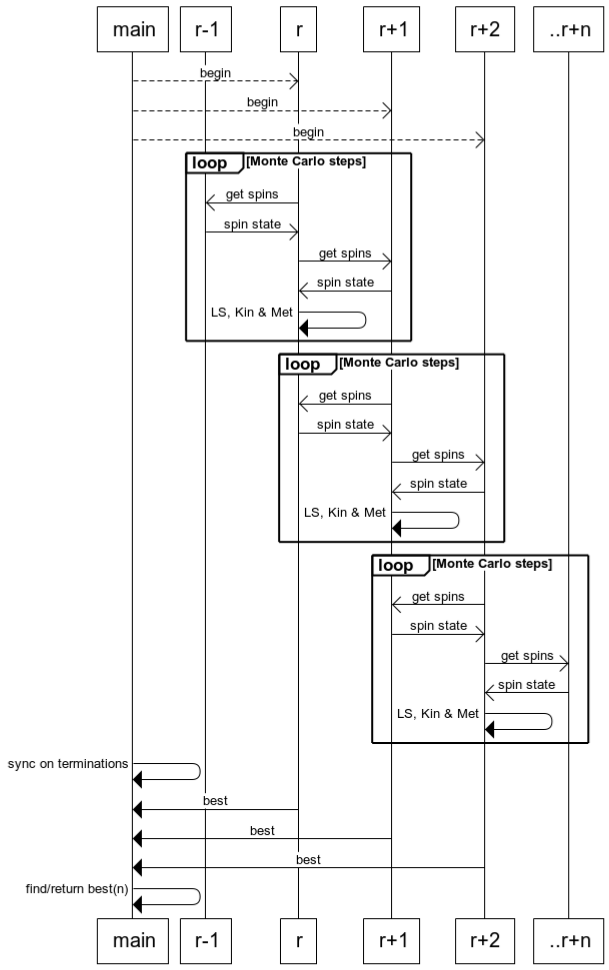


Figure 4.1: Sequence diagram showing a simple thread model: the local search (LS), kinetic calculations (Kin) and Metropolis criteria (Met) are performed for each replica in their own thread, allowing the entire population to be processed in parallel

Figure 4.1 shows how it is possible to parallelize the processing of QA by arranging each replica on its own thread. Replicas asynchronously trade spin status across thread contexts for every private iteration, until the thread has exhausted its Monte Carlo steps. The main thread waits for all replicas to terminate before gathering results and returning the best solution. With this kind of model it is expected that the longer the threads execute, the more they will drift apart in time - some replicas will finish long before others and will only be able communicate their terminal statuses - their spins will appear frozen relative to extant threads. This has implications for the accuracy of the path integral, with the possibility that the solution trajectories (paths) of active replica threads can become biased towards the ones which have stopped. These solutions have ceased to evolve and there is no guarantee that any terminated thread will contain a good enough solution to help guide the optimization which continues in the live threads.

To mitigate this possibility, it is better to place groups of replicas together on each thread, creating subsets of the whole path integral. Running threads still have many active trajectories at hand to compensate for the bias of those which have terminated. Grouping replicas in this manner also reduces the amount of inter-thread communication as only the outermost positioned replicas (first and last of each group) are required to share their spins across execution boundaries. This also decreases the likelihood of any thread being placed into a condition where it is waiting upon spin status information from a neighbouring thread, thus improving performance. Of further benefit is that this scheme decouples the maximum number of replicas from the number of cores or threads a computing system allows, thus improving the scalability of the algorithm. A low-specification computing system may still run a reasonable number of replicas, although at the expense of memory consumption. (With clustered and cloud computing becoming widespread, the likelier prospect is that even larger groups of replicas will be distributed as processes which are managed using load balancing techniques.)

The configuration of the grouping scheme shown in Figure 4.2 is flexible. It may be configured such that a single thread processes a single replica, duplicating the behaviour of Figure 4.1. It is therefore a generalization of the simpler scheme and is the one implemented for use in the experiments in section 4.3.

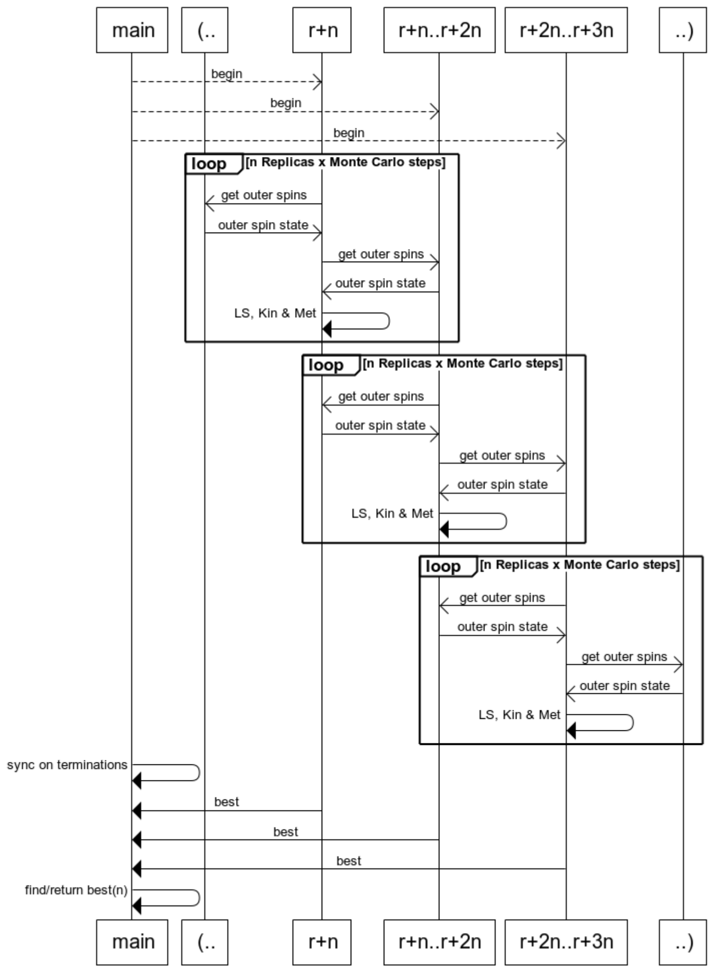


Figure 4.2: Sequence diagram: local search (LS), kinetic calculations (Kin) and Metropolis criteria (Met) are performed upon replicas grouped into threads, allowing population subsets to be processed in parallel

### 4.2.1 Parallelized QA for CVRP (PQAVRP)

The spin encoding and local search schemes for PQAVRP are identical to the single threaded algorithm described in section 2.2 (Figure 2.1) and section 2.3. To recap, the replica is a two-dimensional matrix containing bits which individually describe bi-directional connections between customers. In other words, these are edges which contribute to form an entire route for each vehicle. The total distance of the routes in any one matrix provides the value of the classical potential energy, $H_p$. During the local search phase of the metaheuristic, neighbourhood operators $N$={*Move, Swap, Move-string, Swap-string, 2-Opt, 2-Opt∗*} repeatedly and stochastically flip the state of the bits in the matrix producing a fluctuation in the kinetic energy term $H_k$. Consequently the Hamiltonian is perturbed, which provides an additional mechanism by which the search can escape from local minima within the potential energy landscape of the VRP instance.

**Variables**: $E$: Execution contexts (threads), $g$: group size (replicas per thread), $P$: number of replicas, $T$: temperature, $J_\Gamma$: magnetic coupling strength, $M_C$ : number of Monte Carlo steps, $N$: set of neighbourhood operators, $n$: chosen operator, $S$: set of replicas, $S_{best}$ : best solution, $s'$: candidate solution, $z$: index of current replica.

```
1:  $S \leftarrow$ circular list of feasible randomized solutions
2:  $E \leftarrow$ subsets of $S$, each containing $g$ contiguous replicas
3:  for all $E$ do
4:      Initialize with $\{T, J_\Gamma, M_C, N\}$
5:  end for
6:  for all $E$ do
7:      Begin
8:  end for
9:  while Busy($E$) do
10:     Wait
11: end while
12: return Best($E$)
```

Figure 4.3: Main thread of Parallelized Quantum Annealing.

The pseudo-code of the parallelized algorithm is shown in figure 4.3 which is the main thread, and in figure 4.4, the child thread which processes a subset of the replica list. The main thread is responsible for creating the circular list of replicas and initializing each with a randomized starting solution on line 1. On line 2 child threads are created and each is supplied a subset of the list. Lines 3 to 5 initialize each child thread with a copy of the controlling parameters. The child threads are instructed to begin processing their replica subsets in lines 6 to 8, and the main thread waits for all threads to finish in lines 9 to 11 after which on line 12 it returns the best solution from amongst all the children.

**Variables**: $G$: group of replicas, $g$: number of replicas, $T$: temperature, $J_\Gamma$: magnetic coupling strength, $M_C$ : number of Monte Carlo steps, $N$: set of neighbourhood operators, $n$: chosen operator, $s_{best}$ : best solution, $s'$: candidate solution, $z$: index of current replica, $R_{prev}$: previous replica, $R_{next}$: next replica, $R_{last}$: last replica in previous execution context, $R_{1st}$: first replica in following execution context

```
 1: $s_{best} \leftarrow Best(G)$
 2: **while** $M_C > 0$ **do**
 3:   $z \leftarrow 0$
 4:   **while** $z < g$ **do**
 5:     randomly choose $n \in N$
 6:     $s' \leftarrow n(G_z)$
 7:     $\Delta H_p \leftarrow H_p(s') - H_p(G_z)$
 8:     **if** $z = 0$ **then**
 9:       $R_{prev} \leftarrow Sync(R_{last})$ {Mutex/semaphore}
10:       $R_{next} \leftarrow G_{z+1}$
11:     **else if** $z = g - 1$ **then**
12:       $R_{prev} \leftarrow G_{z-1}$
13:       $R_{next} \leftarrow Sync(R_{1st})$ {Mutex/semaphore}
14:     **else**
15:       $R_{prev} \leftarrow G_{z-1}$
16:       $R_{next} \leftarrow G_{z+1}$
17:     **end if**
18:     $\Delta H_k \leftarrow \text{Interact}(R_{prev}, s', R_{next}) - \text{Interact}(R_{prev}, G_z, R_{next})$
19:     $\Delta H \leftarrow \Delta H_p + J_\Gamma \Delta H_k$
20:     **if** $((\Delta H_p <= 0)$ **or** $(\Delta H <= 0))$ **then**
21:       $G_z \leftarrow s'$
22:     **else if** $\exp(-\Delta H/T) > \text{random}(0, 1)$ **then**
23:       $G_z \leftarrow s'$
24:     **end if**
25:     **if** $H_p(G_z) < H_p(s_{best})$ **then**
26:       $s_{best} \leftarrow G_z$
27:     **end if**
28:     $z \leftarrow z + 1$
29:   **end while**
30:   $M_C \leftarrow M_C - 1$
31: **end while**
32: **return** $s_{best}$
```

Figure 4.4: Child thread of Parallelized Quantum Annealing.

The child thread has many similarities with the generalized algorithm (section 2.3 figure 2.2), sharing the nested loop structure, interaction and acceptance functions, and energy calculations. However, because it only owns a subset of the replica list, it needs to acquire spin states from siblings (existing in their own execution contexts) when processing the first and last replicas in its own list. This difference is shown in lines 8 to 17 where a cross-thread communication technique must be employed to sychronize operation. If the current replica is the first in the subset (line 8), then the state $R_{prev}$ must be acquired from the last replica held in the previous sibling thread (line 9). Conversely, if the current replica is the last in the subset (line 11), $R_{next}$ is acquired from the first replica in the next sibling thread (line 13). In all other cases, $R_{prev}$ and $R_{next}$ can be acquired immediately from within the current execution context (lines 15 and 16).

## 4.3 Experiments

The parallelized algorithm PQAVRP described in section 4.2.1 was implemented in C++ (ISO/IEC 14882:2011) as a multithreaded command-line executable program for use in the Debian GNU/Linux 8 64-bit operating system deployed on a computing platform with 32GB of RAM, and an Intel Xeon E5-2683 v3 processor, having 14 cores and a clock speed of 2GHz.

### 4.3.1 Single-threaded versus Multi-threaded QA algorithms

Tables 4.1 to 4.3 compares the performance of the single-threaded algorithm, FJ-QACVRP [42] with PQAVRP while solving the P benchmarks of Augerat et al [59]. The algorithms were configured identically, with the controlling parameters $\{J_\Gamma, T, P\}$ identified in [42] as resulting in extremely high success rates. For each instance 100 runs were performed and additionally for PQAVRP, repeated with a variety of thread counts $n_M = \{10, 25, 40, 55, 70, 85\}$ chosen to gradually tax the computing platform's resources, and the operating system's ability to supply and effectively maintain threads. Times are shown in seconds, with $t_s$ indicating single-threaded performance of FJ-QACVRP, and $t_M$ the multithreaded PQAVRP. These are the times taken for the optimizers to locate the optimal solution in each instance and is termed $t_{opt}$

Table 4.1 shows the best possible performance of both algorithms i.e. the quickest single run with an optimal solution. In every case, PQAVRP significantly out-performs FJ-QACVRP, requiring at least 15 threads to do so. For P-n55-k10 with $P = 80$, PQAVRP uses 65 threads to execute in 0.03% of the time of FJ-QACVRP. For P-n70-k10 with $P = 60$, 45 threads are needed to find the optimal solution in 7.9% of the time.

Table 4.2 shows the worst case performance while still locating the optimal solutions. The same trend as before is found. In P-n55-k7 with $P = 70$, PQAVRP using 30 threads runs in 0.44% of the time of FJ-QACVRP, while in P-n50-k10 with $P = 80$ uses 75 threads to run in 5.5%.

Table 4.3 shows the average performance of each algorithm to locate optimal solutions. Here, no change in trend is found, with PQAVRP out-performing FJ-QACVRP in every case.

Table 4.1: Set P: Minimum time taken for the single ($t_S$) and multithreaded ($t_M$) optimizers to find the optimal solution for each problem instance. $n_M$ denotes the number of threads used by the multithreaded optimizer. At around 15+ threads, the time for the multithreaded optimizer is an order or more faster.

| | ***P*** | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|
| P-n40-k5 | $t_S$ | $150 \cdot 10^{-3}$ | $170 \cdot 10^{-3}$ | $180 \cdot 10^{-3}$ | $220 \cdot 10^{-3}$ | $200 \cdot 10^{-3}$ | $300 \cdot 10^{-3}$ | $310 \cdot 10^{-3}$ |
| | $t_M$ | $1.6 \cdot 10^{-3}$ | $370 \cdot 10^{-6}$ | $2.9 \cdot 10^{-6}$ | $10 \cdot 10^{-6}$ | $8.4 \cdot 10^{-6}$ | $2.7 \cdot 10^{-6}$ | $320 \cdot 10^{-6}$ |
| | $n_M$ | 40 | 25 | 50 | 60 | 40 | 65 | 25 |
| P-n45-k5 | $t_S$ | $230 \cdot 10^{-3}$ | $240 \cdot 10^{-3}$ | $320 \cdot 10^{-3}$ | $330 \cdot 10^{-3}$ | $360 \cdot 10^{-3}$ | $420 \cdot 10^{-3}$ | $470 \cdot 10^{-3}$ |
| | $t_M$ | $75 \cdot 10^{-6}$ | $640 \cdot 10^{-6}$ | $1.2 \cdot 10^{-3}$ | $350 \cdot 10^{-6}$ | $110 \cdot 10^{-6}$ | $140 \cdot 10^{-6}$ | $6.2 \cdot 10^{-6}$ |
| | $n_M$ | 25 | 40 | 55 | 40 | 55 | 35 | 65 |
| P-n50-k7 | $t_S$ | $350 \cdot 10^{-3}$ | $430 \cdot 10^{-3}$ | $360 \cdot 10^{-3}$ | $540 \cdot 10^{-3}$ | $610 \cdot 10^{-3}$ | $490 \cdot 10^{-3}$ | $820 \cdot 10^{-3}$ |
| | $t_M$ | $9.4 \cdot 10^{-6}$ | $1.3 \cdot 10^{-3}$ | $94 \cdot 10^{-6}$ | $440 \cdot 10^{-6}$ | $1.4 \cdot 10^{-3}$ | $1.0 \cdot 10^{-3}$ | $110 \cdot 10^{-6}$ |
| | $n_M$ | 35 | 40 | 55 | 70 | 60 | 35 | 80 |
| P-n50-k10 | $t_S$ | 3.5 | 1.4 | 1.6 | 3.5 | 7.9 | 5.8 | 4.8 |
| | $t_M$ | $98 \cdot 10^{-3}$ | $150 \cdot 10^{-3}$ | $120 \cdot 10^{-3}$ | $18 \cdot 10^{-3}$ | $32 \cdot 10^{-3}$ | $18 \cdot 10^{-3}$ | $3.9 \cdot 10^{-3}$ |
| | $n_M$ | 15 | 45 | 45 | 70 | 75 | 80 | 70 |
| P-n51-k10 | $t_S$ | $250 \cdot 10^{-3}$ | $170 \cdot 10^{-3}$ | $370 \cdot 10^{-3}$ | $760 \cdot 10^{-3}$ | $670 \cdot 10^{-3}$ | $510 \cdot 10^{-3}$ | $670 \cdot 10^{-3}$ |
| | $t_M$ | $23 \cdot 10^{-3}$ | $6.0 \cdot 10^{-3}$ | $1.6 \cdot 10^{-3}$ | $200 \cdot 10^{-6}$ | $1.2 \cdot 10^{-3}$ | $1.4 \cdot 10^{-3}$ | $1.7 \cdot 10^{-3}$ |
| | $n_M$ | 30 | 35 | 35 | 30 | 75 | 60 | 20 |
| P-n55-k7 | $t_S$ | $580 \cdot 10^{-3}$ | $420 \cdot 10^{-3}$ | $860 \cdot 10^{-3}$ | $830 \cdot 10^{-3}$ | $880 \cdot 10^{-3}$ | $990 \cdot 10^{-3}$ | 1.5 |
| | $t_M$ | $710 \cdot 10^{-6}$ | $3.1 \cdot 10^{-3}$ | $1.6 \cdot 10^{-3}$ | $3.2 \cdot 10^{-3}$ | $6.3 \cdot 10^{-3}$ | $3.3 \cdot 10^{-3}$ | $930 \cdot 10^{-6}$ |
| | $n_M$ | 30 | 40 | 25 | 45 | 60 | 60 | 75 |
| P-n55-k10 | $t_S$ | 1.6 | $700 \cdot 10^{-3}$ | 2.0 | 1.5 | 2.0 | 1.2 | 3.4 |
| | $t_M$ | $42 \cdot 10^{-3}$ | $34 \cdot 10^{-3}$ | $14 \cdot 10^{-3}$ | $6.3 \cdot 10^{-3}$ | $600 \cdot 10^{-6}$ | $1.2 \cdot 10^{-3}$ | $27 \cdot 10^{-3}$ |
| | $n_M$ | 20 | 20 | 45 | 35 | 65 | 60 | 40 |
| P-n60-k10 | $t_S$ | 1.0 | $560 \cdot 10^{-3}$ | $950 \cdot 10^{-3}$ | 1.0 | $770 \cdot 10^{-3}$ | 2.1 | 1.1 |
| | $t_M$ | $9.6 \cdot 10^{-3}$ | $3.4 \cdot 10^{-3}$ | $20 \cdot 10^{-3}$ | $12 \cdot 10^{-3}$ | $18 \cdot 10^{-3}$ | $580 \cdot 10^{-6}$ | $2.1 \cdot 10^{-3}$ |
| | $n_M$ | 40 | 45 | 55 | 45 | 80 | 65 | 65 |
| P-n60-k15 | $t_S$ | $490 \cdot 10^{-3}$ | $320 \cdot 10^{-3}$ | $890 \cdot 10^{-3}$ | $770 \cdot 10^{-3}$ | 1.3 | 1.2 | 1.8 |
| | $t_M$ | $6.3 \cdot 10^{-3}$ | $2.5 \cdot 10^{-3}$ | $7.6 \cdot 10^{-3}$ | $5.4 \cdot 10^{-3}$ | $3.8 \cdot 10^{-3}$ | $3.2 \cdot 10^{-3}$ | $1.3 \cdot 10^{-3}$ |
| | $n_M$ | 30 | 50 | 55 | 55 | 45 | 60 | 70 |
| P-n65-k10 | $t_S$ | $600 \cdot 10^{-3}$ | $690 \cdot 10^{-3}$ | $790 \cdot 10^{-3}$ | 1.2 | 1.2 | $920 \cdot 10^{-3}$ | 1.3 |
| | $t_M$ | $360 \cdot 10^{-6}$ | $2.4 \cdot 10^{-3}$ | $1.6 \cdot 10^{-3}$ | $1.7 \cdot 10^{-3}$ | $700 \cdot 10^{-6}$ | $6.1 \cdot 10^{-3}$ | $3.5 \cdot 10^{-3}$ |
| | $n_M$ | 35 | 35 | 55 | 40 | 25 | 70 | 35 |
| P-n70-k10 | $t_S$ | 1.8 | 2.6 | 1.4 | 1.4 | 3.5 | 2.5 | 3.0 |
| | $t_M$ | $41 \cdot 10^{-3}$ | $190 \cdot 10^{-3}$ | $110 \cdot 10^{-3}$ | $5.9 \cdot 10^{-3}$ | $13 \cdot 10^{-3}$ | $13 \cdot 10^{-3}$ | $94 \cdot 10^{-3}$ |
| | $n_M$ | 30 | 30 | 45 | 20 | 60 | 55 | 25 |
| P-n76-k4 | $t_S$ | 4.4 | 8.4 | 17 | 5.2 | 10 | 8.2 | 5.5 |
| | $t_M$ | $470 \cdot 10^{-3}$ | $140 \cdot 10^{-3}$ | $310 \cdot 10^{-3}$ | $130 \cdot 10^{-3}$ | $72 \cdot 10^{-3}$ | $350 \cdot 10^{-3}$ | $190 \cdot 10^{-3}$ |
| | $n_M$ | 20 | 30 | 50 | 15 | 75 | 60 | 30 |
| P-n76-k5 | $t_S$ | 5.1 | 3.4 | 3.9 | 3.3 | 3.9 | 3.1 | 6.8 |
| | $t_M$ | $210 \cdot 10^{-3}$ | $170 \cdot 10^{-3}$ | $140 \cdot 10^{-3}$ | $140 \cdot 10^{-3}$ | $43 \cdot 10^{-3}$ | $160 \cdot 10^{-3}$ | $100 \cdot 10^{-3}$ |
| | $n_M$ | 20 | 45 | 30 | 65 | 65 | 50 | 40 |
| P-n101-k4 | $t_S$ | 9.7 | 8.6 | 12 | 15 | 23 | 16 | 18 |
| | $t_M$ | $320 \cdot 10^{-3}$ | $220 \cdot 10^{-3}$ | $390 \cdot 10^{-3}$ | $480 \cdot 10^{-3}$ | $370 \cdot 10^{-3}$ | $640 \cdot 10^{-3}$ | $650 \cdot 10^{-3}$ |
| | $n_M$ | 40 | 20 | 30 | 35 | 65 | 15 | 65 |

Table 4.2: Set P: Maximum time time taken for the single ($t_S$) and multithreaded ($t_M$) optimizers to find the optimal solution for each problem instance. $n_M$ denotes the number of threads used by the multithreaded optimizer. At 10+ threads, the time for the multithreaded optimizer is an order or more faster.

| | $\boldsymbol{P}$ | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|
| | $t_S$ | $700\cdot10^{-3}$ | $550\cdot10^{-3}$ | $840\cdot10^{-3}$ | $820\cdot10^{-3}$ | $810\cdot10^{-3}$ | $970\cdot10^{-3}$ | $830\cdot10^{-3}$ |
| P-n40-k5 | $t_M$ | $11\cdot10^{-3}$ | $21\cdot10^{-3}$ | $15\cdot10^{-3}$ | $18\cdot10^{-3}$ | $17\cdot10^{-3}$ | $20\cdot10^{-3}$ | $20\cdot10^{-3}$ |
| | $n_M$ | 30 | 15 | 25 | 30 | 35 | 40 | 55 |
| | $t_S$ | 2.3 | 1.5 | 2.2 | 2.2 | 2.0 | 2.0 | 1.3 |
| P-n45-k5 | $t_M$ | $16\cdot10^{-3}$ | $19\cdot10^{-3}$ | $19\cdot10^{-3}$ | $36\cdot10^{-3}$ | $20\cdot10^{-3}$ | $33\cdot10^{-3}$ | $40\cdot10^{-3}$ |
| | $n_M$ | 40 | 35 | 55 | 60 | 35 | 45 | 30 |
| | $t_S$ | 4.0 | 12 | 8.1 | 10 | 7.3 | 7.7 | 6.2 |
| P-n50-k7 | $t_M$ | $72\cdot10^{-3}$ | $60\cdot10^{-3}$ | $54\cdot10^{-3}$ | $82\cdot10^{-3}$ | $100\cdot10^{-3}$ | $150\cdot10^{-3}$ | $160\cdot10^{-3}$ |
| | $n_M$ | 35 | 20 | 30 | 40 | 55 | 40 | 75 |
| | $t_S$ | 780 | 850 | 720 | 890 | 550 | $1.2\cdot10^{3}$ | 720 |
| P-n50-k10 | $t_M$ | 36 | 44 | 32 | 27 | 30 | 43 | 36 |
| | $n_M$ | 30 | 45 | 40 | 20 | 75 | 10 | 70 |
| | $t_S$ | 180 | 120 | 40 | 55 | 89 | 39 | 60 |
| P-n51-k10 | $t_M$ | 1.4 | $630\cdot10^{-3}$ | $630\cdot10^{-3}$ | $720\cdot10^{-3}$ | $340\cdot10^{-3}$ | $570\cdot10^{-3}$ | $730\cdot10^{-3}$ |
| | $n_M$ | 20 | 30 | 55 | 15 | 30 | 25 | 80 |
| | $t_S$ | 58 | 160 | 93 | 120 | 94 | 210 | 73 |
| P-n55-k7 | $t_M$ | $610\cdot10^{-3}$ | 1.1 | 1.0 | $530\cdot10^{-3}$ | $590\cdot10^{-3}$ | 1.0 | $450\cdot10^{-3}$ |
| | $n_M$ | 25 | 20 | 15 | 30 | 70 | 65 | 40 |
| | $t_S$ | 730 | 820 | $1.1\cdot10^{3}$ | 840 | 980 | $1.6\cdot10^{3}$ | $1.1\cdot10^{3}$ |
| P-n55-k10 | $t_M$ | 29 | 23 | 29 | 34 | 23 | 24 | 29 |
| | $n_M$ | 20 | 20 | 45 | 70 | 50 | 70 | 40 |
| | $t_S$ | 55 | 60 | 87 | 57 | 63 | 71 | 36 |
| P-n60-k10 | $t_M$ | 1.2 | 1.3 | 1.4 | 1.4 | $880\cdot10^{-3}$ | 1.9 | 1.2 |
| | $n_M$ | 35 | 35 | 20 | 65 | 55 | 80 | 40 |
| | $t_S$ | 240 | 420 | 390 | 410 | 240 | 230 | 400 |
| P-n60-k15 | $t_M$ | 3.6 | 5.8 | 7.2 | 5.7 | 5.8 | 2.1 | 3.0 |
| | $n_M$ | 30 | 40 | 40 | 50 | 50 | 55 | 45 |
| | $t_S$ | 19 | 48 | 60 | 24 | 50 | 38 | 41 |
| P-n65-k10 | $t_M$ | $340\cdot10^{-3}$ | $490\cdot10^{-3}$ | $700\cdot10^{-3}$ | $330\cdot10^{-3}$ | $560\cdot10^{-3}$ | $200\cdot10^{-3}$ | $550\cdot10^{-3}$ |
| | $n_M$ | 20 | 20 | 25 | 40 | 80 | 55 | 50 |
| | $t_S$ | 620 | 440 | 910 | 390 | 400 | 390 | 410 |
| P-n70-k10 | $t_M$ | 12 | 9.9 | 10 | 12 | 11 | 12 | 8.3 |
| | $n_M$ | 20 | 45 | 20 | 70 | 60 | 70 | 50 |
| | $t_S$ | $1.0\cdot10^{3}$ | $1.6\cdot10^{3}$ | 930 | 860 | 760 | $1.0\cdot10^{3}$ | $1.0\cdot10^{3}$ |
| P-n76-k4 | $t_M$ | 14 | 23 | 22 | 31 | 29 | 26 | 24 |
| | $n_M$ | 35 | 50 | 35 | 15 | 55 | 80 | 65 |
| | $t_S$ | $1.5\cdot10^{3}$ | $1.8\cdot10^{3}$ | $1.1\cdot10^{3}$ | $1.7\cdot10^{3}$ | $2.4\cdot10^{3}$ | $1.1\cdot10^{3}$ | $2.7\cdot10^{3}$ |
| P-n76-k5 | $t_M$ | 43 | 44 | 29 | 39 | 28 | 46 | 43 |
| | $n_M$ | 15 | 30 | 35 | 50 | 40 | 70 | 80 |
| | $t_S$ | 360 | 380 | 330 | 310 | 240 | 530 | 300 |
| P-n101-k4 | $t_M$ | 8.9 | 9.7 | 12 | 11 | 9.6 | 10 | 10 |
| | $n_M$ | 25 | 35 | 50 | 55 | 65 | 35 | 50 |

Table 4.3: Set P: Average time time taken for the single ($t_S$) and multithreaded ($t_M$) optimizers to find the optimal solution for each problem instance. $n_M$ denotes the number of threads used by the multithreaded optimizer. At 10+ threads, the time for the multithreaded optimizer is an order or more faster.

| | $\boldsymbol{P}$ | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|
| | $t_S$ | $290\cdot10^{-3}$ | $310\cdot10^{-3}$ | $390\cdot10^{-3}$ | $440\cdot10^{-3}$ | $460\cdot10^{-3}$ | $520\cdot10^{-3}$ | $540\cdot10^{-3}$ |
| P-n40-k5 | $t_M$ | $6.4\cdot10^{-3}$ | $8.6\cdot10^{-3}$ | $9.1\cdot10^{-3}$ | $9.4\cdot10^{-3}$ | $9.0\cdot10^{-3}$ | $12\cdot10^{-3}$ | $12\cdot10^{-3}$ |
| | $n_M$ | 30 | 50 | 25 | 30 | 40 | 65 | 55 |
| | $t_S$ | $450\cdot10^{-3}$ | $460\cdot10^{-3}$ | $560\cdot10^{-3}$ | $630\cdot10^{-3}$ | $720\cdot10^{-3}$ | $780\cdot10^{-3}$ | $820\cdot10^{-3}$ |
| P-n45-k5 | $t_M$ | $8.2\cdot10^{-3}$ | $12\cdot10^{-3}$ | $11\cdot10^{-3}$ | $14\cdot10^{-3}$ | $13\cdot10^{-3}$ | $18\cdot10^{-3}$ | $21\cdot10^{-3}$ |
| | $n_M$ | 30 | 20 | 55 | 55 | 35 | 60 | 35 |
| | $t_S$ | 1.5 | 1.8 | 1.6 | 1.9 | 2.0 | 2.1 | 2.3 |
| P-n50-k7 | $t_M$ | $27\cdot10^{-3}$ | $24\cdot10^{-3}$ | $23\cdot10^{-3}$ | $19\cdot10^{-3}$ | $41\cdot10^{-3}$ | $47\cdot10^{-3}$ | $52\cdot10^{-3}$ |
| | $n_M$ | 35 | 20 | 55 | 40 | 55 | 60 | 60 |
| | $t_S$ | 130 | 150 | 120 | 160 | 130 | 160 | 150 |
| P-n50-k10 | $t_M$ | 8.6 | 11 | 7.2 | 6.4 | 7.5 | 5.5 | 7.0 |
| | $n_M$ | 40 | 45 | 60 | 20 | 80 | 15 | 80 |
| | $t_S$ | 13 | 12 | 8.9 | 9.2 | 10 | 8.6 | 8.7 |
| P-n51-k10 | $t_M$ | $210\cdot10^{-3}$ | $210\cdot10^{-3}$ | $110\cdot10^{-3}$ | $93\cdot10^{-3}$ | $100\cdot10^{-3}$ | $140\cdot10^{-3}$ | $200\cdot10^{-3}$ |
| | $n_M$ | 30 | 30 | 45 | 35 | 30 | 25 | 15 |
| | $t_S$ | 12 | 12 | 10 | 10 | 10 | 13 | 7.9 |
| P-n55-k7 | $t_M$ | $64\cdot10^{-3}$ | $190\cdot10^{-3}$ | $150\cdot10^{-3}$ | $170\cdot10^{-3}$ | $190\cdot10^{-3}$ | $190\cdot10^{-3}$ | $170\cdot10^{-3}$ |
| | $n_M$ | 10 | 25 | 55 | 45 | 70 | 60 | 75 |
| | $t_S$ | 180 | 180 | 150 | 160 | 160 | 170 | 200 |
| P-n55-k10 | $t_M$ | 5.9 | 2.5 | 4.3 | 5.3 | 3.8 | 5.4 | 5.1 |
| | $n_M$ | 40 | 20 | 45 | 25 | 65 | 70 | 70 |
| | $t_S$ | 14 | 11 | 14 | 14 | 13 | 15 | 13 |
| P-n60-k10 | $t_M$ | $230\cdot10^{-3}$ | $400\cdot10^{-3}$ | $340\cdot10^{-3}$ | $430\cdot10^{-3}$ | $370\cdot10^{-3}$ | $500\cdot10^{-3}$ | $320\cdot10^{-3}$ |
| | $n_M$ | 40 | 35 | 20 | 65 | 80 | 60 | 65 |
| | $t_S$ | 53 | 46 | 55 | 47 | 41 | 38 | 35 |
| P-n60-k15 | $t_M$ | $570\cdot10^{-3}$ | $900\cdot10^{-3}$ | $510\cdot10^{-3}$ | $590\cdot10^{-3}$ | $490\cdot10^{-3}$ | $340\cdot10^{-3}$ | $430\cdot10^{-3}$ |
| | $n_M$ | 35 | 40 | 40 | 50 | 45 | 55 | 45 |
| | $t_S$ | 6.3 | 7.1 | 8.1 | 6.6 | 7.6 | 7.1 | 7.2 |
| P-n65-k10 | $t_M$ | $87\cdot10^{-3}$ | $130\cdot10^{-3}$ | $190\cdot10^{-3}$ | $83\cdot10^{-3}$ | $170\cdot10^{-3}$ | $70\cdot10^{-3}$ | $180\cdot10^{-3}$ |
| | $n_M$ | 20 | 15 | 25 | 40 | 65 | 55 | 45 |
| | $t_S$ | 74 | 78 | 93 | 97 | 84 | 99 | 90 |
| P-n70-k10 | $t_M$ | 3.3 | 3.5 | 3.4 | 2.8 | 3.5 | 4.2 | 3.7 |
| | $n_M$ | 30 | 45 | 20 | 20 | 60 | 15 | 50 |
| | $t_S$ | 190 | 230 | 210 | 200 | 220 | 220 | 250 |
| P-n76-k4 | $t_M$ | 5.9 | 8.1 | 7.2 | 9.4 | 10 | 11 | 9.3 |
| | $n_M$ | 35 | 20 | 35 | 55 | 70 | 80 | 30 |
| | $t_S$ | 250 | 330 | 270 | 310 | 240 | 250 | 310 |
| P-n76-k5 | $t_M$ | 12 | 14 | 8.1 | 13 | 12 | 14 | 14 |
| | $n_M$ | 15 | 45 | 35 | 50 | 25 | 25 | 70 |
| | $t_S$ | 83 | 82 | 94 | 96 | 100 | 97 | 110 |
| P-n101-k4 | $t_M$ | 3.2 | 4.7 | 4.0 | 4.4 | 4.7 | 3.9 | 4.9 |
| | $n_M$ | 25 | 25 | 15 | 65 | 35 | 35 | 30 |

Within the chosen limits of a population size restricted to $P \leq 100$ occupying at most 2.5x of the maximum available number processor cores, the performance of PQAVRP is exceptional when compared to FJ-QACVRP. PQAVRP returns the same success rate and solution quality as the single-threaded predecessor but in a significantly and consistently smaller fraction of time.

### 4.3.2 Multithreaded performance

Figure 4.5 shows the effect of increasing the thread count $n_M$ upon $t_{opt}$. When $n_M = 1$, the performance approximates that of FJ-QACVRP. $t_{opt}$ diminishes as threads are added, with the best improvement being in the worst-case performance - the longest time (max) to find optimal solutions. As can be expected, when the number of threads begins to exceed the number of cores, improvements in time begin to tail off - this can be seen to happen at around 14 threads where the curves start to become shallow. Beyond 60 threads, average (av) and best times (min) do not improve indicating that computer system is becoming saturated with the processes.

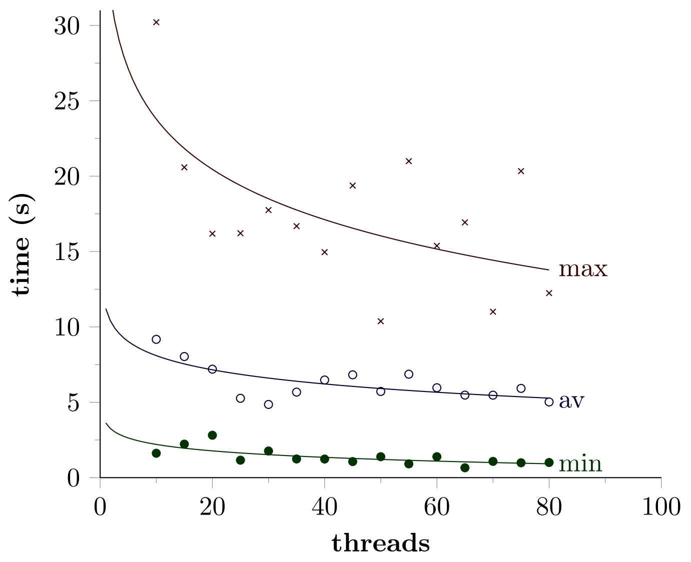


Figure 4.5: Number of threads versus maximum (*max*), average (*av*), and minimum (*min*) time until the optimal solution is found for P-n101-k4 with P=100.

## 4.4 Conclusions

This section has been shown that for restricted running times, the multithreaded variant is able to outperform the single-threaded QA optimizers in a comprehensive fashion. For a thread-count which does not exceed the computing capabilities of the host platform, the coarse-grained parallelized program was proven to deliver identical solution quality and equivalent or improved success rates in a fraction of the wall-clock time. Additionally, since the original formulation of the QA algorithm remains intact, employing PIMC and the Ising model, parameter tunings from previous studies for each VRP instance could be reused. It was shown that for all the benchmarks in this study, no retuning was required. Whether or not this situation holds true for all VRP benchmarks is the subject for future experimentation.

With the multithreaded advantages of QA proven, it should now be possible to refactor the program code for the replica threads, enabling them to work as a distinct process in a distributed manner such as in clustered or cloud computing. This would mitigate the limitations of an individual computing system which is easily overloaded by specifying an excessive number of threads. Such technologies present additional opportunities for future investigation.

Chapter 5

# The Industrial Domain of Field Service Scheduling

## 5.1 Introduction

Knowledge Transfer Partnerships (Knowledge Transfer Partnership (KTP)) [76] are organized by Innovate UK [77], a body charged by the UK government to help businesses innovate and grow. An academic organization, a graduate, and a company are partnered to their mutual benefits whilst they transfer knowledge from academia, converting it to practical and economic advantage.

Owing to the exposure of the MMU's research into optimization of scheduling problems [41], and the attractiveness of the Innovate UK programme, ServicePower Technologies Plc initiated a KTP with the aim to update their Field Service Scheduling (FSS) technology with a more effective, patented design based upon QA. Materials for and outputs of the KTP can be found in Appendix B Knowledge Transfer Partnership Outcomes.

The KTP for applying QA to FSS was begun after the completion of the research outlined in Chapter 2 which established the basis of the QA algorithm, and mid-way through the tuning experiments of Chapter 3. This offered the opportunity to extend the experimental regime already in place for VRP to include FSS. As well as furthering the applicability of the algorithm, it might confirm the wider applicability of the PT and ESPT methodologies.

The agreed development plan was to gradually enhance the product *Simulated Annealing for Service Optimization* (Simulated Annealing for Service Optimization (SASO)) over several stages (Figure 5.1) so that the behaviour of the software in application frameworks would remain consistent with previous revisions. In broad terms, first the SASO data structures were modified so that it could host a population of solutions, each processed by the existing SA metaheuristic. Once working, the Quasi-Quantum Annealing (QQA) metaheuristic - which employed a random kinetic energy calculation instead of an Ising Model - was added to the program. Comparative experiments were conducted to assess the performance of both metaheuristics.

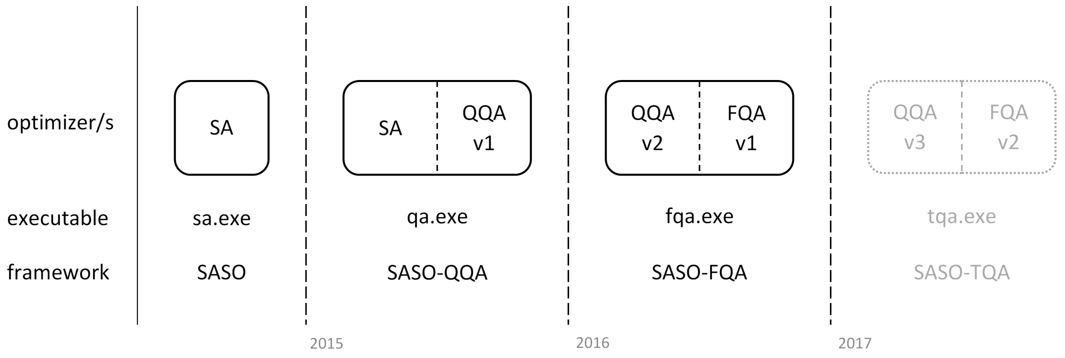


Figure 5.1: The evolution of ServicePower optimizers. Quasi-Quantum Annealing was integrated with Simulated Annealing in 2015, with the latter being supplanted in 2016 by Full Quantum Annealing. Multithreaded variants of the quantum algorithm were introduced in 2017 and onwards.

The second stage removed any trace of SA, and then included a spin encoding scheme for FSS (section 5.4), implementing the Ising Model with which kinetic energy could be calculated. This would be the first version of QA for FSS and is referred to here as Full Quantum Annealing (FQA). Again, experiments were conducted to compare the metaheuristics. The third stage involved using multithreading techniques to parallelize QQA and FQA. This was an optional stage, development of which depended upon whether performance of FQA was sufficient for use in production environments.

The major contribution in this chapter is the encoding scheme for FSS which is central to implementing the Ising model. It is to our knowledge the first spin representation of a such a complex scheduling problem, the novelty of which is affirmed by the award of a patent. Also a key contribution is the QAFSS algorithm - the first time QA has been applied to solve both academic and practical real-world instances of FSS problems. Further contributions in this chapter are the parallelized FQA algorithm PQAFSS which improves the run-time performance of QAFSS, and the pared-down QA algorithm Quasi-Quantum Annealing which proved to be more than just a stepping-stone along the way to FQA, demonstrating itself as an effective metaheuristic in its own right.

Section 5.3 details the stages of the development plan, and section 5.5 presents the results of the performance experiments. The appendices contain the outputs of the KTP programme.

## 5.2 Field Service Scheduling

In FSS problems, the individuals of a workforce must each be assigned a sequence of jobs from within a schedule which is formed to service the needs of a geographically distributed collection of customers. Field Service industries send their workforce to sites where they perform maintenance, emergency or otherwise, upon industrial and domestic machinery; or to deliver and install equipment; or to perform instrumentation and measurement tasks. The workforce is a set of heterogeneous workers with attributes such as skills, spares and capacity, location, mobility and availability (shift pattern, vacation).

The customers have varying appointment preferences and requirements which define the kinds of jobs which need to be assigned to the workforce. Each customer has a demand upon spares stock and the type of skill needed to fulfill their requirements. The objective function is usually a combination of several factors which must be minimized. For example: overtime; mean tardiness; service level agreement fines; total travel time; etc.

FSS combines features of other scheduling problems such as in time tabling, rostering, job-shop, and routing problems such as the Traveling Technician Problem (TTP) [78], and the Technician Routing and Scheduling Problem (TRSP) [79]. FSS can be tackled as a dynamic problem in which workforce and customer sets may vary continuously in size (jobs are booked throughout hours of business, whilst workers can alter shifts) and attributes such as travel time (affected by traffic or weather conditions) can affect jobs assignments in real time. The work presented here tackles the static variant of FSS, wherein appointment times are endeavoured to be kept, shift patterns are known in advance and do not change for the given scheduling horizon, and other variable attributes are assumed ideal e.g. spares stock is limitless (return to depots are factored into travel times).

## 5.3 Development of QA for FSS

The basis of the QA algorithm for FSS (QAFSS) was established in Chapter 2, where it was applied to the Vehicle Routing Problem (VRP) and subsequently extended and modified to accomplish simplified tuning of the controlling parameters in Chapter 3. Wall-clock time performance issues were then addressed by parallelizing the algorithm, taking advantage of modern multi-core processors and multithreaded environments in Chapter 4. With many of the drawbacks of QA now addressed, application in an industrial domain was feasible. The VRP and FSS problem domains were simultaneously addressed during the latter efforts.

### 5.3.1 Simulated and Quasi-Quantum Annealing for Field Service Scheduling

All of the optimizers in the ServicePower product Optimization on Demand (OoD) were evolved from an implementation of SA for FSS within an application framework entitled SASO. The program structure of the first version Quasi-Quantum Annealing (QQA) resembled coupled or ensemble-based SA [80][81], reusing a large portion of the SA code base. It was therefore able to co-exist within the same executable and be hosted in a slightly modified SASO framework. The domain logic, which included objectives and a large set of constraints to encode the business rules, was unchanged as it was necessary that existing infrastructure was reused.

### 5.3.2 Full Quantum Annealing for Field Service Scheduling

The development of Full Quantum Annealing (FQA) shared nothing but the domain model with SA. QQA structures and methods were upgraded to incorporate an Ising Model (section 5.4) for FQA whilst all traces of SA were removed. Other upgrades included objectifying structures, enhancing performance, removing bugs, adding features to increase robustness and support experiments. Since FQA is a generalization of QQA, they co-exist in the same application - improvements applied to any one are naturally acquired by both. Owing to these improvements, their version numbers were increased (QQAv2, FQAv2).

### 5.3.3 Parallelized Quantum Annealing for Field Service Scheduling

For convenience and expediency, the *SASO* framework also hosted the multithreaded QA optimizers, allowing a resuse of the experimental framework for previous optimizers (SA/QQA/FQA) requiring only small adjustments and additions to the tools, scripts and code.

## 5.4 QA Scheme for FSS

The QA algorithm for FSS is broadly identical to figure 3.4 in Chapter 3 as used in FJ-QACVRP, employing the PIMC method and Metropolis criteria, while also presenting the opportunity to be tuned using ESPT (Chapter 3,section 3.2.3). The major differences are in the definition of the spin matrix and the implementation of a set of neighbourhood operators which are tailored specifically for local searching of scheduling problems. The SASO neighbourhood operators and local search functions were reused in QAFSS, leaving only the spin matrix to be defined.

### 5.4.1 Spin Matrix for FSS

A spin matrix representation of a FSS problem must be defined in order to form the Ising Model which is used by the QA algorithm. Elements of the schedule which must be minimized (or maximized) are reduced to +1,-1 spin values and then mapped into a matrix of Boolean values. Figure 5.2 shows the encoding scheme, which can be described as a table whose rows are assignments of jobs $j_0 - j_1$ to operatives $O_0 - O_n$ (service technicians) over the scheduling window which is divided into periods $t_0 - t_n$. Hard and soft constraint violations and conflicts in the schedule are tracked in $h_0 - h_n$, $s_0 - s_n$, and $c_0 - c_n$ respectively. As the local search scheme progresses, jobs are moved stochastically amongst operatives, breaking and satisfying constraints whilst conflicts are caused and then resolved. These changes cause the cells of each row to flip their state $\{0, 1\}$ continually during optimization, and so cause perturbations in the kinetic energy when this arrangement is used in the calculation of interaction energy (Chapter 2, figure 2.3). A large number of these matrices are used to populate a circular list and so forms the Ising Model for QAFSS. A conceptual view of the spin-spin interactions is shown in Figure 5.3.

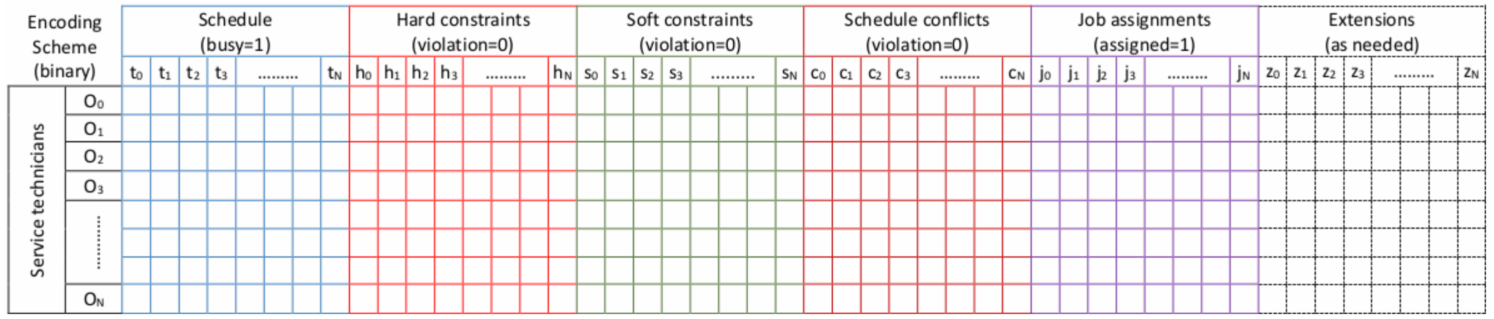

Encoding Scheme (binary)
Schedule (busy=1)
Hard constraints (violation=0)
Soft constraints (violation=0)
Schedule conflicts (violation=0)
Job assignments (assigned=1)
Extensions (as needed)
$t_0$ $t_1$ $t_2$ $t_3$ ......... $t_N$
$h_0$ $h_1$ $h_2$ $h_3$ ......... $h_N$
$s_0$ $s_1$ $s_2$ $s_3$ ......... $s_N$
$c_0$ $c_1$ $c_2$ $c_3$ ......... $c_N$
$j_0$ $j_1$ $j_2$ $j_3$ ......... $j_N$
$z_0$ $z_1$ $z_2$ $z_3$ ......... $z_N$
Service technicians
$O_0$
$O_1$
$O_2$
$O_3$
..........
$O_N$

Figure 5.2: Encoding scheme which transforms a schedule into matrix of spins. In memory, this is defined as an array of Boolean variables which is contained in each replica object.

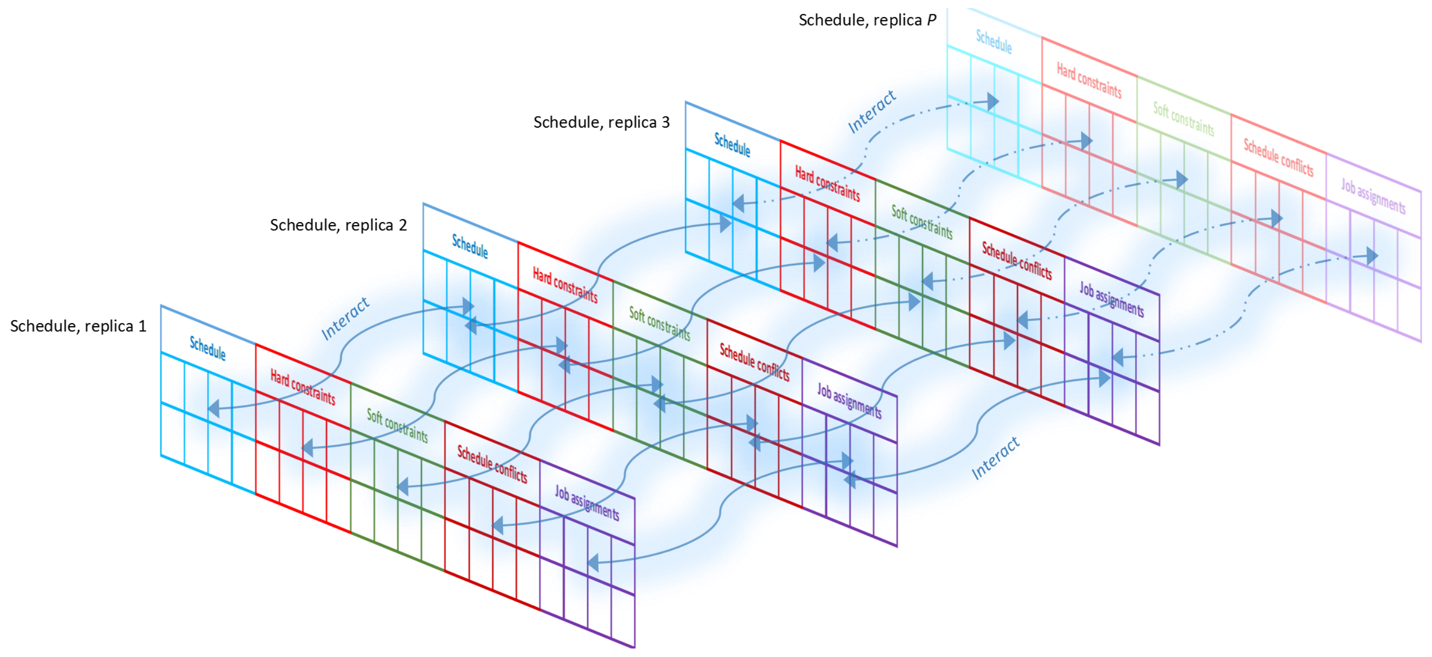


Figure 5.3: Conceptual view of the Quantum Annealing process at a single iteration. The interaction sites of the Ising Model are the cells of the schedule encoded as a spin matrix. These specify the aspects of the problem which are to be optimized for. The interactions between corresponding spins of neighbouring replicas help simulate the phenomenon of quantum tunnelling

Table 5.1: Example scenario. Four operatives with skills and locations, to be assigned four jobs subject to two hard and two soft constraints

| **Service Technicians** | **Skills** | **Location** |
|---|---|---|
| $O_0$ | A | $L_A$ |
| $O_1$ | ABC | $L_B$ |
| $O_2$ | CD | $L_C$ |
| $O_3$ | ABCD | $L_D$ |

| **Hard constraint** | **Description** |
|---|---|
| $h_0$ | start on time |
| $h_1$ | correct skills |

| **Soft constraint** | **Description** |
|---|---|
| $s_0$ | overtime |
| $s_1$ | location |

| **Job** | **Start period** | **Duration** | **Skills needed** | **Location** |
|---|---|---|---|---|
| $j_0$ | 0 | 2 | A | $L_A$ |
| $j_1$ | 1 | 1 | AB | $L_A$ |
| $j_2$ | 1 | 3 | B | $L_D$ |
| $j_3$ | 2 | 2 | C | $L_C$ |

Table 5.2: Example spin matrix for the scenario in Table 5.1. Four technicians $O_0$-$O_3$ with various combinations of skills $ABCD$, located at $L_A$-$L_D$ must be assigned jobs $j_0$-$j_3$ whilst obeying constraints $h_0$-$h_1$ and $s_0$-$s_1$. As shown, there exists a boolean encoding of job assignments to service technicians over a quartered time period having no hard constraint violations and a single soft constraint violation with no conflicts (e.g. double-bookings)

| | Periods | | | | Constraints | | | | Conflicts | | | | Assignments | | | | Hexadecimal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $t_0$ | $t_1$ | $t_2$ | $t_3$ | $h_0$ | $h_1$ | $s_0$ | $s_1$ | $c_0$ | $c_1$ | $c_2$ | $c_3$ | $j_0$ | $j_1$ | $j_2$ | $j_3$ | encoding |
| $O_0$ | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | CFF8 |
| $O_1$ | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 4EF4 |
| $O_2$ | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 3FF1 |
| $O_3$ | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 7FF2 |

### 5.4.2 Parallelized QA for FSS

The previous versions of the ServicePower metaheuristics employed single-threaded implementations of QA. As this is a population-based method, wherein each member of the population can be viewed as a distinct optimization process, it is clear why it is slower than SA - a trajectory method which runs in a single process. QA has a wall-clock time which is approximately equal to that of SA multiplied by a factor of the population size. In order to reduce this factor, several approaches may be taken but only a few would leave enough of the algorithm intact so that it may still be described faithfully as 'quantum'. Straight-forward code optimizations are one such method but opportunities for improvements are rapidly exhausted and simultaneously subject to diminishing returns relative to the effort of implementation. Parallelization using threading techniques is a one-off approach which in principle allows performance gains which are scaled by the availability of computational resources. The designs shown in figures 4.1 and 4.2 of section 4.2 leverage this technique and were implemented for FSS. The Simple (S) and Grouped (G) threading models were hosted together in the same program, PQAFSS, and were included in further comparative experiments (section 5.5.3).

## 5.5 Experiments and comparative results

Although multicore processor architectures allow for programs to be written to achieve parallelization of their workload, QQA and FQA are not implemented to support concurrent processing of replicas. However, the experimental scripts take advantage in a different way. Rather than breaking the algorithm into parallelizable components, QQA and FQA were invoked by a script instructing the operating system to place a whole experimental run in a separate process which becomes allocated to a single CPU core. This design allows runs to occur simultaneously, and maximize core usage whilst delivering results quickly. Two CPUs were employed (Table 5.3), one which resides locally in a workstation, and the other hosted in the cloud, each running the Linux operating system.

Table 5.3: Processors used for QQA versus SA experiments.

| **Manufacturer** | **Processor** | **Cores** | **Clock (GHz)** | **Host** | **Key** |
|---|---|---|---|---|---|
| Intel | Xeon E5-2683 v3 | 14 | 2.0 | Workstation | X1 |
| Intel | Xeon E5-2666 v3 | 32 | 2.9 | AWS Cloud | X2 |

### 5.5.1 Quasi-Quantum versus Simulated Annealing, Initial results

A collection of synthetic instances (Table 5.4) was used in the experimental comparisons, with levels of complexity ranging from trivial to, by academic standards, modestly complex. The simplest of these proved useful during the development, testing and debugging of the software whilst the largest give indications of the performance levels that can be expected for more complicated data sets, and motivation to explore possibilities to optimize the software for speed.

Table 5.4: Instances used for QQA versus SA experiments

| **Name** | **#Jobs** | **#Operatives** | **Notes** |
|---:|:---:|:---:|:---:|
| gen10-3 | 10 | 3 | Useful for debugging/testing |
| gen22 | 20 | 5 | SA & QA results equivalent in cost |
| gen50 | 50 | 5 | |
| gen50-10rnd | 50 | 10 | Randomized job durations |
| gen100 | 100 | 10 | Under-resourced version of gen100-15 |
| gen100-15 | 100 | 15 | - |
| gen500 | 500 | 50 | - |

For the last four instances in Table 5.4, a batch of runs was repeated for a variety of iteration values $M_C$ (Monte Carlo steps). Each batch included runs at a range of temperatures ($1 \leq T \leq 1001$, $\Delta T = 100$) and for QA, at each temperature interval a range of replica counts ($10 \leq P \leq 50$, $\Delta P = 10$). For statistical significance, 100 runs were performed at each interval, and so each batch consists of an instance which is solved over 5000 times.

Tables 5.5 to 5.8 present the objective function values and wall-clock times for the best and worst solutions found in each batch by the optimizers.

Table 5.5: Objective function values (OF) for instance gen50-10rnd (50 jobs, 10 operatives), using a restricted number of Monte Carlo steps. SA performs well, delivering results in less than a second. Quasi-quantum annealing (QQA) is able to perform equally in the best case when given a population size of 10, albeit slower. In the worst case, QQA outperforms SA but requires at least 20 replicas and more time to do so.

| | | $M_C$=50 | | $M_C$=100 | |
|---|---|---|---|---|---|
| | | OF | sec | OF | sec |
| SA | best | 78620[1] | <1 | 78620 | <1 |
| | worst | 86390 | <1 | 80465 | <1 |
| QQA | best | 78620[2] | 4 | 78620[2] | 8 |
| | worst | 83420[3] | 9 | 78620[4] | 43 |

[1] $T$=201
[2] $T$=101, $P$=10
[3] $T$=901, $P$=20
[4] $T$=601, $P$=50

Table 5.6: Objective function values (OF) for instance gen100

| | | $M_C$=5000 | |
|---|---|---|---|
| | | OF | sec |
| SA | best | 1067300[1] | 91 |
| | worst | 1069150 | 91 |
| QQA | best | 1066310[2] | 1121 |
| | worst | 1067430[3] | 921 |

[1] $T$=101
[2] $T$=901, $P$=20
[3] $T$=801, $P$=10

Table 5.7: Objective function values (OF) for instance gen100-15

| | | $M_C$=50 | | $M_C$=1000 | | $M_C$=5000 | |
|---|---|---|---|---|---|---|---|
| | | OF | sec | OF | sec | OF | sec |
| SA | best | 32760 | 1 | 31040[1] | 13 | 31050 | 65 |
| | worst | 43740 | 1 | 31590 | 13 | 31600 | 66 |
| QQA | best | 31640 | 21 | 30930[2] | 554 | 30930 | 666 |
| | worst | 44280 | 12 | 31250[3] | 269 | 31140 | 2837 |

[1] $T$=501
[2] $T$=701, $P$=40
[3] $T$=1, $P$=20

Table 5.8: Objective function values (OF) for instance gen500

| | | $M_C$=5000 | |
|---|---|---|---|
| | | OF | sec |
| SA | best | 5099850[1] | 26 |
| | worst | 5101970 | 27 |
| QQA | best | 5098510[2] | 655 |
| | worst | 5100570[3] | 284 |

[1] $T$=501
[2] $T$=801, $P$=40
[3] $T$=201, $P$=10

In terms of best scores, QQA outperforms SA in nearly all instances. Where it does not, the score is equivalent and obtained at lower values for $M_C$ (Table 5.5, $M_C \leq 100$). Regarding worst scores, QQA generally outperforms SA except in two instances and when iterations are at the lowest (tables 5.5 and 5.7, $M_C$=50). This may indicate that SA is able to converge in fewer steps than QQA which lacks the additional convergence supplied by a proper kinetic energy component as in QA. Convergence appears to improve when $M_C \geq 100$, whereupon SA is consistently outperformed.

In terms of wall-clock time SA completes faster than QQA. This is unsurprising as SA is a trajectory method, in effect having only a single replica to process. QQA for the same number of iterations should take approximately $\times P$ more time than SA. This is bourne out in the findings which show the correct order of magnitude if not this precise relationship. For example, taking the best solutions in Table 5.6, QA averages 56 seconds per replica and SA 91 seconds, and for the worst QA averages 92.1 seconds per replica and SA 91 seconds.

Tables 5.9 and 5.10 present the mininum, maximum and average counts for unresourced jobs. This is a contextual measure important in many use-cases as it expresses congestion in the schedule owing to unavailability of assignees or an overloading in jobs. It can also express a lack of performance in the optimizer. Pinpointing why is difficult because there are many potential causes, such as inadequate tuning, problems in the local search, or not enough allotted time or iterations.

Table 5.9: Unresourced job counts for instance gen50-10rnd

| | $M_C$=50 | | | $M_C$=100 | | |
|---|---|---|---|---|---|---|
| | min | max | Av | min | max | Av |
| SA | 20 | 22 | 20.32 | 21 | 23 | 21.3 |
| QQA | 20 | 22 | 20.24 | 20 | 21 | 20.4 |

Table 5.10: Unresourced job counts for instance gen100-15

| | $M_C$=50 | | | $M_C$=1000 | | | $M_C$=5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | min | max | Av | min | max | Av | min | max | Av |
| SA | 0 | 2 | 0.4 | 0 | 0 | 0 | 0 | 0 | 0 |
| QQA | 0 | 2 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 |

Tables 5.9 and 5.10 show that at low iterations ($M_C$=50), SA and QQA are almost equivalent with only the averages being divergent. In the first case QQA is slightly better, in the second it is marginally worse. At twice the iterations (Table 5.9, $M_C$=100) QQA shows a definite improvement compared to SA, with a lower minimum, maximum and average number of unresourced jobs.

### 5.5.2 Full Quantum versus Quasi-Quantum versus Simulated Annealing

The first version of the ServicePower application, SASO, employed the SA metaheuristic as the optimization technique for FSS problems. Further development produced a second application which employed both full-blown and simplified versions the QA metaheuristic. These versions were entitled Full Quantum Annealing (FQA) and Quasi-Quantum Annealing (QQA) respectively. With all three metaheuristics complete and deployed using the same supporting infrastructure it is possible to configure them similarly and provide identical experimental conditions to compare their performance.

Table 5.11 shows the instances used in this comparison, where the data was sourced, and indicates the complexity of each problem (number of jobs and resources to be assigned within a scheduling window).

Table 5.11: Instances used in comparative experiments of FQA, QQA and SA

| Instance | Source | Jobs | Resources | Scheduling window (days) |
|---:|---|---|---|---|
| gen500 | Synthetic | 500 | 50 | 1 |
| gen1000-90 | Synthetic | 1000 | 90 | 1 |
| gen5k-600 | Synthetic | 5000 | 600 | 1 |
| gen10k-12k | Synthetic | 10000 | 12000 | 1 |
| aXXXXXXX_3d | Real world | 2429 | 379 | 3 |
| aXXXXXXX_05012016 | Real world | 1045 | 127 | 1 |
| aDXXXXX_05012016 | Real world | 63 | 31 | 1 |
| cXXXXXXX_05012016 | Real world | 301 | 87 | 1 |
| hXX_XXXXXX_04012016 | Real world | 363 | 21 | 1 |
| OPXXXXXXXXXXXX5d | Unknown | 13361 | 9100 | 154 |
| OPXXXXXXXXXXXX15d | Unknown | 4962 | 899 | 15 |
| OPXXXXXXXXXXXX_05012016 | Unknown | 536 | 75 | 1 |
| sXXXX_8107_F2_3day | Real world | 1966 | 339 | 3 |
| sXXXX_8107_F2_23122015 | Real world | 1115 | 171 | 1 |
| uXXXX4 | Real world | 1761 | 296 | 7 |
| uXXXX4x0.75R | Real world | 1761 | 222 | 7 |
| uXXXX4x0.50R | Real world | 1761 | 148 | 7 |
| uXXXX4x0.25R | Real world | 1761 | 75 | 7 |

The method used to rank solutions is shown in Figure 5.4. Computation time is measured in units of program iterations, and solution quality is defined using the following prioritized metrics, with ties broken in highest (1) to lowest (4) order:

1. Most jobs assigned
2. Least resources used
3. Weighted sum of constraint violations and business costs i.e. The *Objective Function.*
4. Wall-clock time

Figure 5.4: Solution quality defined as a prioritized list of metrics

#### 5.5.2.1 Untuned optimization

The summarized results in Table 5.12 best represents the approach to tuning that occurs often out in the field. Currently, the SA optimizer is deployed with little or no tuning i.e. Temperature values are not systematically chosen. They are either best-guessed or copied from previous deployments and reused. Therefore, in these experiments the optimizers are given the same temperature value while the Quantum variants also share population sizes and magnetic field values which are estimated from experience. As can be seen, FQA accumulates the most wins, whereas SA is only able to compete on the basis of time and only when the other quality metrics are tied.

Table 5.12: Summary of results using untuned optimizers SA, QQA and FQA

| Instance | Best optimizer | | |
|---|---|---|---|
| | Most jobs assigned | Least resources used | Solution Quality |
| gen500 | tie | tie | SA |
| gen1000-90 | FQA | tie | FQA |
| gen5k-600 | tie | FQA | FQA |
| gen10k-12k | tie | FQA | FQA |
| aXXXXXXX_3d | tie | QQA | QQA |
| aXXXXXXX_05012016 | QQA | FQA | QQA |
| aDXXXXX_05012016 | tie | tie | SA |
| cXXXXXXX_05012016 | tie | FQA | FQA |
| hXX_XXXXXX_04012016 | tie | tie | QQA |
| OPXXXXXXXXXXXX5d | FQA | FQA | FQA |
| OPXXXXXXXXXXX15d | FQA | QQA | FQA |
| OPXXXXXXXXXXXX_05012016 | tie | FQA | FQA |
| sXXXX_8107_F2_3day | FQA | QQA | FQA |
| sXXXX_8107_F2_23122015 | tie | FQA | FQA |
| uXXXX4 | QQA | tie | QQA |
| uXXXX4x0.75R | FQA | tie | FQA |
| uXXXX4x0.50R | QQA | FQA | QQA |
| uXXXX4x0.25R | FQA | QQA | FQA |
| Total (SA : QQA : FQA : tie) | 0 : 3 : 6 : 9 | 0 : 4 : 8 : 6 | 2 : 5 : 11 : 0 |

For every instance in the study, each untuned optimizer was run for $16.8 \times 10^6$ iterations. In total, tables 5.13 and 5.14 represent 54 runs of the software.

Table 5.13: Results of the untuned optimizers upon 1-9 problem instances (part 1 of 2)

| Instance | | SA | QQA | FQA |
|---|---|---|---|---|
| gen500 | $J_{UR}$ | 0 | 0 | 0 |
| | $R_{US}$ | 0 | 0 | 0 |
| | $O_F$ | 5131050 | 5123490 | 5124220 |
| | $t$ | 30 | 843 | 1015 |
| gen1000-90 | $J_{UR}$ | 248 | 244 | 241 |
| | $R_{US}$ | 0 | 0 | 0 |
| | $O_F$ | 988660 | 969920 | 958065 |
| | $t$ | 31 | 844 | 1008 |
| gen5k-600 | $J_{UR}$ | 0 | 0 | 0 |
| | $R_{US}$ | 7 | 5 | 8 |
| | $O_F$ | 51507260 | 51497040 | 51492700 |
| | $t$ | 39 | 1281 | 1439 |
| gen10k-12k | $J_{UR}$ | 0 | 0 | 0 |
| | $R_{US}$ | 0 | 0 | 2 |
| | $O_F$ | 4137680 | 4223330 | 4115420 |
| | $t$ | 46 | 1584 | 1938 |
| aXXXXXXX_3d | $J_{UR}$ | 181 | 107 | 107 |
| | $R_{US}$ | 213 | 217 | 215 |
| | $O_F$ | 845025456 | 449241102 | 449419784 |
| | $t$ | 23 | 544 | 693 |
| aXXXXXXX_05012016 | $J_{UR}$ | 110 | 58 | 60 |
| | $R_{US}$ | 47 | 47 | 48 |
| | $O_F$ | 397847727 | 245301213 | 250715470 |
| | $t$ | 23 | 570 | 734 |
| aDXXXXX_05012016 | $J_{UR}$ | 0 | 0 | 0 |
| | $R_{US}$ | 10 | 10 | 10 |
| | $O_F$ | 25985 | 25985 | 25985 |
| | $t$ | 12 | 243 | 285 |
| cXXXXXXX_05012016 | $J_{UR}$ | 41 | 35 | 35 |
| | $R_{US}$ | 37 | 35 | 36 |
| | $O_F$ | 124989130 | 108068305 | 107650425 |
| | $t$ | 17 | 408 | 509 |
| hXX_XXXXXX_04012016 | $J_{UR}$ | 2 | 0 | 0 |
| | $R_{US}$ | 4 | 4 | 4 |
| | $O_F$ | 7847522 | 600584 | 600667 |
| | $t$ | 55 | 1525 | 1779 |

Table 5.14: Results of the untuned optimizers upon 10-18 problem instances (part 2 of 2)

| Instance | | SA | QQA | FQA |
|---|---|---|---|---|
| OPXXXXXXXXXXXX5d | $J_{UR}$ | 2090 | 1899 | 1809 |
| | $R_{US}$ | 8727 | 8636 | 8706 |
| | $O_F$ | 7175576615 | 6586513060 | 6397401870 |
| | $t$ | 40 | 972 | 1218 |
| OPXXXXXXXXXXXX15d | $J_{UR}$ | 44 | 30 | 22 |
| | $R_{US}$ | 390 | 417 | 391 |
| | $O_F$ | 1696531935 | 1629601385 | 1635615975 |
| | $t$ | 44 | 911 | 1006 |
| OPXXXXXXXXXXXX_05012016 | $J_{UR}$ | 33 | 33 | 33 |
| | $R_{US}$ | 27 | 25 | 27 |
| | $O_F$ | 99053905 | 99053345 | 99052585 |
| | $t$ | 33 | 667 | 820 |
| sXXXX_8107_F2_3day | $J_{UR}$ | 13 | 7 | 5 |
| | $R_{US}$ | 114 | 127 | 126 |
| | $O_F$ | 105396292 | 81960904 | 77542484 |
| | $t$ | 23 | 588 | 747 |
| sXXXX_8107_F2_23122015 | $J_{UR}$ | 12 | 3 | 3 |
| | $R_{US}$ | 47 | 49 | 50 |
| | $O_F$ | 73598290 | 45411775 | 45231710 |
| | $t$ | 24 | 604 | 743 |
| uXXXX4 | $J_{UR}$ | 27 | 6 | 7 |
| | $R_{US}$ | 10 | 10 | 10 |
| | $O_F$ | 1796910 | 1213255 | 1188230 |
| | $t$ | 26 | 707 | 882 |
| uXXXX4x0.75R | $J_{UR}$ | 533 | 514 | 503 |
| | $R_{US}$ | 8 | 7 | 7 |
| | $O_F$ | 6161480 | 5991755 | 5982740 |
| | $t$ | 29 | 767 | 906 |
| uXXXX4x0.50R | $J_{UR}$ | 955 | 914 | 931 |
| | $R_{US}$ | 5 | 5 | 7 |
| | $O_F$ | 10084960 | 9745425 | 9827700 |
| | $t$ | 34 | 873 | 1179 |
| uXXXX4x0.25R | $J_{UR}$ | 1373 | 1355 | 1349 |
| | $R_{US}$ | 3 | 5 | 3 |
| | $O_F$ | 13973780 | 13798740 | 13762710 |
| | $t$ | 46 | 1329 | 1613 |

#### 5.5.2.2 Tuned optimization

The summary of Table 5.15 presents the results of the tuned metaheuristics over a range of iterations upon a selection of the problem instances. Although it is the fastest algorithm, SA shows no wins as it is always outperformed in quality metrics 1, 2 and 3. Ties are shown to indicate that FQA and QQA performed identically in metrics 1, 2 and 3. If the quality metric is strictly applied then QQA wins in these cases because it is a slightly faster algorithm than FQA and the tie can be broken using metric 4, wall-clock time.

Table 5.15: Summary of results using tuned optimizers QQA and FQA

| Instance | Best solution quality over $N \times 10^6$ iterations | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $N$ | 0.5 | 1 | 1.5 | 2.5 | 5 | 10 | 20 | 40 |
| aXXXXXXX_3d | FQA | QQA | QQA | QQA | FQA | QQA | FQA | FQA |
| aXXXXXXX_05012016 | FQA | QQA | tie | QQA | FQA | tie | FQA | tie |
| cXXXXXXX_05012016 | tie | QQA | tie | FQA | tie | QQA | FQA | FQA |
| hXX_XXXXXX_04012016 | tie | tie | tie | tie | tie | tie | tie | tie |
| sXXXX_8107_F2_3day | QQA | QQA | FQA | FQA | FQA | FQA | QQA | QQA |
| sXXXX_8107_F2_23122015 | FQA | QQA | tie | QQA | FQA | FQA | FQA | tie |
| uXXXX4 | QQA | QQA | QQA | QQA | QQA | QQA | FQA | QQA |
| uXXXXXx0.75R | FQA | SA | SA | FQA | FQA | QQA | FQA | SA |
| uXXXXXx0.50R | QQA | QQA | FQA | FQA | FQA | FQA | FQA | FQA |
| uXXXXXx0.25R | QQA | QQA | FQA | QQA | FQA | QQA | FQA | FQA |

For brevity, a sample of the tuned results is presented in sections 5.5.2.3 to 5.5.2.5 below. Over 3,600 executions of the software were performed, in various configurations of temperature, population size and magnetic field. In each configuration, runs occurred across a range of iterations.

(Not shown is any data from which the tuning values were determined as there were over 37,000 software executions - a copious amount that could not easily be presented here. Unless otherwise stated, control parameters were obtained from experimental suites which used $5\times 10^6$ iterations per execution.)

#### 5.5.2.3 aXXXXXX_3d

According to solution quality (Figure 5.4), Table 5.16 shows that FQA and QQA have equal numbers of wins, with FQA performing better at high iterations.

Table 5.16: Tuned optimizers with increasing iterations scheduling 2429 jobs to 379 resources in a 1-day window

| Iterations $\times 10^6$ | | SA | QQA | FQA |
|---|---|---|---|---|
| 0.5 | $J_{UR}$ | 270 | 253 | 251 |
| | $R_{US}$ | 200 | 211 | 212 |
| | $O_F$ | 1414076988 | 912009082 | 909912860 |
| | $t$ | 1 | 9 | 12 |
| 1 | $J_{UR}$ | 248 | 209 | 209 |
| | $R_{US}$ | 208 | 211 | 209 |
| | $O_F$ | 1103110993 | 789607771 | 745025145 |
| | $t$ | 1 | 19 | 23 |
| 1.5 | $J_{UR}$ | 235 | 189 | 192 |
| | $R_{US}$ | 210 | 213 | 210 |
| | $O_F$ | 1252516913 | 709683151 | 699090774 |
| | $t$ | 2 | 28 | 24 |
| 2.5 | $J_{UR}$ | 215 | 166 | 166 |
| | $R_{US}$ | 212 | 217 | 213 |
| | $O_F$ | 964641643 | 632257554 | 623862294 |
| | $t$ | 3 | 45 | 59 |
| 5 | $J_{UR}$ | 189 | 136 | 133 |
| | $R_{US}$ | 208 | 215 | 215 |
| | $O_F$ | 1100053108 | 535865682 | 526027990 |
| | $t$ | 7 | 91 | 114 |
| 10 | $J_{UR}$ | 182 | 120 | 120 |
| | $R_{US}$ | 210 | 217 | 212 |
| | $O_F$ | 869283167 | 488956120 | 487477361 |
| | $t$ | 14 | 177 | 229 |
| 20 | $J_{UR}$ | 170 | 104 | 101 |
| | $R_{US}$ | 214 | 216 | 217 |
| | $O_F$ | 833127551 | 440775590 | 432125823 |
| | $t$ | 28 | 353 | 460 |
| 40 | $J_{UR}$ | 160 | 92 | 90 |
| | $R_{US}$ | 211 | 217 | 214 |
| | $O_F$ | 783653192 | 405086421 | 399130440 |
| | $t$ | 56 | 756 | 939 |

#### 5.5.2.4 cXXXXXXX_05012016

Table 5.17 shows that QQA attains best solution quality 5 times versus 3 for FQA. Again, FQA obtains best results at higher iterations.

Table 5.17: Tuned optimizers with increasing iterations scheduling assign 301 jobs to 87 resources in a 1-day window

| Iterations $\times 10^6$ | | SA | QQA | FQA |
|---|---|---|---|---|
| 0.5 | $J_{UR}$ | 39 | 35 | 35 |
| | $R_{US}$ | 34 | 37 | 37 |
| | $O_F$ | 120154235 | 108252695 | 108372975 |
| | $t$ | 1 | 14 | 18 |
| 1 | $J_{UR}$ | 39 | 35 | 35 |
| | $R_{US}$ | 35 | 38 | 37 |
| | $O_F$ | 120535450 | 107978870 | 108185030 |
| | $t$ | 1 | 28 | 36 |
| 1.5 | $J_{UR}$ | 39 | 35 | 35 |
| | $R_{US}$ | 37 | 37 | 37 |
| | $O_F$ | 119785050 | 108185075 | 108368805 |
| | $t$ | 2 | 42 | 53 |
| 2.5 | $J_{UR}$ | 39 | 35 | 35 |
| | $R_{US}$ | 36 | 37 | 38 |
| | $O_F$ | 119487550 | 108368470 | 108068650 |
| | $t$ | 3 | 70 | 89 |
| 5 | $J_{UR}$ | 39 | 35 | 35 |
| | $R_{US}$ | 36 | 37 | 37 |
| | $O_F$ | 119484660 | 108185260 | 108669930 |
| | $t$ | 5 | 139 | 182 |
| 10 | $J_{UR}$ | 38 | 35 | 35 |
| | $R_{US}$ | 35 | 37 | 36 |
| | $O_F$ | 118468542 | 108195970 | 108251120 |
| | $t$ | 10 | 277 | 363 |
| 20 | $J_{UR}$ | 39 | 35 | 35 |
| | $R_{US}$ | 36 | 37 | 37 |
| | $O_F$ | 119668785 | 108185755 | 108067420 |
| | $t$ | 21 | 552 | 707 |
| 40 | $J_{UR}$ | 39 | 35 | 35 |
| | $R_{US}$ | 36 | 37 | 38 |
| | $O_F$ | 120029600 | 108068390 | 107768230 |
| | $t$ | 43 | 1134 | 1449 |

#### 5.5.2.5 sXXXX_8107_F2_23122015

Table 5.18 shows FQA with best solution quality 5 times versus 3 for QQA. At $40\times 10^6$ iterations, QQA and FQA perform similarly with less than 2% difference in $O_F$.

Table 5.18: Tuned optimizers with increasing iterations scheduling 1115 jobs to 171 resources in a 1-day window

| Iterations $\times 10^6$ | | SA | QQA | FQA |
|---|---|---|---|---|
| 0.5 | $J_{UR}$ | 83 | 53 | 46 |
| | $R_{US}$ | 39 | 48 | 44 |
| | $O_F$ | 283073255 | 193798160 | 171796945 |
| | $t$ | 1 | 20 | 24 |
| 1 | $J_{UR}$ | 53 | 22 | 26 |
| | $R_{US}$ | 45 | 48 | 54 |
| | $O_F$ | 195884665 | 105884835 | 115036305 |
| | $t$ | 2 | 39 | 50 |
| 1.5 | $J_{UR}$ | 39 | 13 | 13 |
| | $R_{US}$ | 38 | 52 | 52 |
| | $O_F$ | 150498705 | 76347540 | 74995515 |
| | $t$ | 2 | 58 | 78 |
| 2.5 | $J_{UR}$ | 18 | 5 | 6 |
| | $R_{US}$ | 43 | 53 | 54 |
| | $O_F$ | 91135620 | 53666135 | 54875005 |
| | $t$ | 4 | 103 | 127 |
| 5 | $J_{UR}$ | 11 | 3 | 3 |
| | $R_{US}$ | 40 | 54 | 56 |
| | $O_F$ | 73291115 | 47370425 | 47915700 |
| | $t$ | 7 | 202 | 269 |
| 10 | $J_{UR}$ | 8 | 3 | 3 |
| | $R_{US}$ | 45 | 57 | 59 |
| | $O_F$ | 63230200 | 44829065 | 45660950 |
| | $t$ | 15 | 409 | 525 |
| 20 | $J_{UR}$ | 6 | 3 | 3 |
| | $R_{US}$ | 37 | 58 | 59 |
| | $O_F$ | 57313540 | 44983080 | 44618800 |
| | $t$ | 30 | 813 | 1100 |
| 40 | $J_{UR}$ | 5 | 3 | 3 |
| | $R_{US}$ | 46 | 59 | 59 |
| | $O_F$ | 53656815 | 44438470 | 45211535 |
| | $t$ | 59 | 1638 | 2188 |

#### 5.5.2.6 uXXXX4

Table 5.19 shows that QQA is well suited for this dataset, with the best solution quality 6 times.

Table 5.19: Tuned optimizers with increasing iterations scheduling 1761 jobs to 296 resources in a 7-day window

| Iterations $\times 10^6$ | | SA | QQA | FQA |
|---|---|---|---|---|
| 0.5 | $J_{UR}$ | 375 | 371 | 370 |
| | $R_{US}$ | 10 | 10 | 10 |
| | $O_F$ | 5930450 | 5749190 | 5804485 |
| | $t$ | 1 | 15 | 18 |
| 1 | $J_{UR}$ | 375 | 360 | 370 |
| | $R_{US}$ | 10 | 10 | 10 |
| | $O_F$ | 5406200 | 5178565 | 5309265 |
| | $t$ | 2 | 29 | 36 |
| 1.5 | $J_{UR}$ | 379 | 367 | 371 |
| | $R_{US}$ | 10 | 10 | 10 |
| | $O_F$ | 5132110 | 5096870 | 5019600 |
| | $t$ | 3 | 44 | 55 |
| 2.5 | $J_{UR}$ | 382 | 368 | 370 |
| | $R_{US}$ | 10 | 10 | 11 |
| | $O_F$ | 4963825 | 4856715 | 4834895 |
| | $t$ | 4 | 70 | 83 |
| 5 | $J_{UR}$ | 378 | 362 | 367 |
| | $R_{US}$ | 14 | 15 | 16 |
| | $O_F$ | 4772705 | 4634295 | 4658180 |
| | $t$ | 8 | 139 | 178 |
| 10 | $J_{UR}$ | 357 | 337 | 348 |
| | $R_{US}$ | 16 | 23 | 26 |
| | $O_F$ | 4505600 | 4260580 | 4340785 |
| | $t$ | 15 | 288 | 353 |
| 20 | $J_{UR}$ | 336 | 307 | 300 |
| | $R_{US}$ | 18 | 28 | 25 |
| | $O_F$ | 4247865 | 3935130 | 3857235 |
| | $t$ | 31 | 560 | 722 |
| 40 | $J_{UR}$ | 271 | 206 | 217 |
| | $R_{US}$ | 19 | 18 | 18 |
| | $O_F$ | 3630550 | 2966555 | 3078100 |
| | $t$ | 66 | 1208 | 1439 |

**Caveat:** Owing to time and resource limitations, the values of control parameters for the uXXXX4 dataset were determined from an experimental suite which used only $0.5\times 10^6$ iterations per run and is therefore possible that convergence of the algorithms did not occur. Were this the case, the values of the control parameters would likely be suboptimal and their usage in the tuned experiments could result in poor performance. A further experimental suite to tune the algorithms using using $10\times 10^6$ iterations or more would be needed to confirm this or to improve control parameter values.

### 5.5.3 Parallelized Quantum Annealing versus single-threaded optimizers

The results in section 5.5.2 presented a comparison of three single-threaded optimizers for solving Field Service Scheduling problems. Overall, QQA and FQA outperformed SA, producing higher quality schedules for the same number of program iterations but their running times were much longer - approximately the number of replicas used multiplied by the time taken by SA. This motivated the development of two multithreaded variants of QA entitled the Simple (S) and Grouped (G) models. Each model divides the processing of the replicas amongst different threads of execution and so reduce the running time by the number of threads. In the following experiments, both models are compared against the single-threaded optimizers using a the data sets in previous experiments. As in section 5.5.2.1, all metaheuristics are executed in their untuned state, and the controlling parameters for FQA are assigned to the S and G models in PQA. Since wall-clock performance is at issue, the running time of the S and G models are restricted to the durations of the best previous times of SA and FQA.

#### 5.5.3.1 Time bounded by SA

Table 5.20 shows the performance of the G and S models when limited to the same running time as SA (shown underlined) which terminated, as did FQA and QQA, after $16.8\times 10^6$ iterations as in the experiments in section 5.5.2, the results of which are repeated here for convenience.

Table 5.20: Results of S and G models limited to the running time of Simulated Annealing

| Data set | | Single threaded | | | Multithreaded | |
|---|---|---|---|---|---|---|
| | | SA | QQA | FQA | S | G |
| | $J_{UR}$ | 0 | 0 | 0 | 0 | 0 |
| gen5k-600 | $R_{US}$ | 7 | 5 | 8 | 46 | 57 |
| | $t$ | **39** | 1281 | 1439 | 39 | 39 |
| | $J_{UR}$ | 181 | 107 | 107 | 161 | 162 |
| aXXXXXXX_3d | $R_{US}$ | 213 | 217 | 215 | 207 | 208 |
| | $t$ | **23** | 544 | 693 | 23 | 23 |
| | $J_{UR}$ | 110 | 58 | 60 | 62 | 59 |
| aXXXXXXX_05012016 | $R_{US}$ | 47 | 47 | 48 | 48 | 47 |
| | $t$ | **23** | 570 | 734 | 22.83 | 23 |
| | $J_{UR}$ | 41 | 35 | 35 | 35 | 35 |
| cXXXXXXX_05012016 | $R_{US}$ | 37 | 35 | 36 | 38 | 38 |
| | $t$ | **17** | 408 | 509 | 11.69 | 10.03 |
| | $J_{UR}$ | 2 | 0 | 0 | 0 | 0 |
| hXX_XXXXXX_04012016 | $R_{US}$ | 4 | 4 | 4 | 4 | 4 |
| | $t$ | **55** | 1525 | 1779 | 0.37 | 0.34 |
| | $J_{UR}$ | 33 | 33 | 33 | 32 | 30 |
| OPXXXXXXXXXXXX_05012016 | $R_{US}$ | 27 | 25 | 27 | 23 | 25 |
| | $t$ | **33** | 667 | 820 | 24.8 | 8.77 |
| | $J_{UR}$ | 13 | 7 | 5 | 7 | 8 |
| sXXXX_8107_F2_3day | $R_{US}$ | 114 | 127 | 126 | 113 | 120 |
| | $t$ | **23** | 588 | 747 | 22.86 | 23 |
| | $J_{UR}$ | 12 | 3 | 3 | 3 | 3 |
| sXXXX_8107_F2_23122015 | $R_{US}$ | 47 | 49 | 50 | 57 | 56 |
| | $t$ | **24** | 604 | 743 | 23.74 | 23.5 |
| | $J_{UR}$ | 533 | 514 | 503 | 479 | 486 |
| uXXXX4x0.75R | $R_{US}$ | 8 | 7 | 7 | 7 | 7 |
| | $t$ | **29** | 767 | 906 | 29 | 29 |
| | $J_{UR}$ | 955 | 914 | 931 | 904 | 905 |
| uXXXX4x0.50R | $R_{US}$ | 5 | 5 | 7 | 5 | 5 |
| | $t$ | **34** | 873 | 1179 | 34 | 34 |
| | $J_{UR}$ | 1373 | 1355 | 1349 | 1340 | 1351 |
| uXXXX4x0.25R | $R_{US}$ | 3 | 5 | 3 | 4 | 3 |
| | $t$ | **46** | 1329 | 1613 | 44.35 | 46 |

In all but one case, both models allocate more jobs (the primary metric $J_{UR}$ which is the number of jobs left unresourced in the backlog, or "intray"; lower is better) than SA and so the secondary metric of unused resources ($R_{US}$ the number of operatives/resources which are unscheduled; higher is better) is never needed as a tie-breaker.

The most notable aspect of these results is in the amount of running time needed to comfortably improve upon or compete with the established single-threaded optimizers; in some cases better results than SA were obtained much sooner than the set time limit. For example, in OPXXXXXXXXXXXX_05012016 only 27% the time limit was required, and in hsb_NEFSS7_04012016, it was less than 1%.

Also notable is that in several cases, the multithreaded optimizers unexpectedly improved upon the results of QQA and FQA which were allowed to run for far longer. The G-Model outperforms FQA in the primary metric $J_{UR}$ for aXXXXXXX_05012016, OPXXXXXXXXXXXX_05012016, uXXXX4x0.75R and uXXXX4x0.50R. Both models also outperform in sXXXX_8107_F2_23122015 and cXXXXXXX_05012016 where the secondary metric $R_{US}$ breaks the tie in the primary.

#### 5.5.3.2 Time bounded by FQA

Table 5.21 shows the performance of the G and S models when the running time is increased to that of FQA (shown underlined) which terminated after $16.8 \times 10^6$ iterations as in section 5.5.2.

Table 5.21: Results of S and G models limited to the running time of Full Quantum Annealing

| Data set | | Single threaded | | | Multithreaded | |
|---|---|---|---|---|---|---|
| | | SA | QQA | FQA | S | G |
| gen5k-600 | $J_{UR}$ | 0 | 0 | 0 | 0 | 0 |
| | $R_{US}$ | 7 | 5 | 8 | 115 | 116 |
| | $t$ | 39 | 1281 | **1439** | 1432 | 1343 |
| aXXXXXXX_05012016 | $J_{UR}$ | 110 | 58 | 60 | 57 | 57 |
| | $R_{US}$ | 47 | 47 | 48 | 48 | 48 |
| | $t$ | 23 | 570 | **734** | 467 | 295 |
| OPXXXXXXXXXXXX_05012016 | $J_{UR}$ | 33 | 33 | 33 | 33 | 33 |
| | $R_{US}$ | 27 | 25 | 27 | 31 | 30 |
| | $t$ | 33 | 667 | **820** | 473 | 714 |
| sXXXX_8107_F2_3day | $J_{UR}$ | 13 | 7 | 5 | 6 | 6 |
| | $R_{US}$ | 114 | 127 | 126 | 111 | 130 |
| | $t$ | 23 | 588 | **747** | 672 | 512 |
| sXXXX_8107_F2_23122015 | $J_{UR}$ | 12 | 3 | 3 | 3 | 3 |
| | $R_{US}$ | 47 | 49 | 50 | 57 | 59 |
| | $t$ | 24 | 604 | **743** | 742 | 719 |
| uXXXX4x0.75R | $J_{UR}$ | 533 | 514 | 503 | 423 | 423 |
| | $R_{US}$ | 8 | 7 | 7 | 9 | 7 |
| | $t$ | 29 | 767 | **906** | 900 | 902 |
| uXXXX4x0.50R | $J_{UR}$ | 955 | 914 | 931 | 825 | 847 |
| | $R_{US}$ | 5 | 5 | 7 | 5 | 6 |
| | $t$ | 34 | 873 | **1179** | 1169 | 1176 |
| uXXXX4x0.25R | $J_{UR}$ | 1373 | 1355 | 1349 | 1273 | 1271 |
| | $R_{US}$ | 3 | 5 | 3 | 3 | 3 |
| | $t$ | 46 | 1329 | **1613** | 1610 | 1609 |

The results in some cases are suggestive that the allowed time is excessive; modest improvements in schedule quality indicate that convergence towards minima has occurred and the optimizers are seeking diminishing returns. For example, $J_{UR}$ in OPXXXXXXXXXXXX_05012016 is unchanged across all optimizers and the secondary metric is necessary to break ties. In other cases, the results for all the QA optimizers are similar. In sXXXX_8107_F2_3day, sXXXX_8107_F2_23122015 and aXXXXXXX_05012016 there are small variations in the primary and secondary metrics which is indicative that the optimizers are searching intensively for tiny improvements from positions located about a local miminum.

In the remaining cases, the multi-threaded variants are clearly converging much later and are thus able to locate larger improvements in each quality metric. In gen5k-600, they are able to save more resources and in the uXXXX sets, schedule between 7% and 18% more jobs.

### 5.5.4 Conclusions

It has been demonstrated that for restricted running times, the multithreaded variants of QA are able to outperform any of the single-threaded optimizers. It has also been shown that for prolonged running times they are able deeply scour the space of possible schedules and discover additional improvements in either the primary or secondary metrics. These results appear encouraging but some reservations need to be considered:

1. The restricted running times may unfairly disadvantage SA since it may not be converge soon enough to return a decent schedule. By virtue of maintaining many concurrent schedules, the S and G models are able view search space through a larger-volumed frustrum and so have a natural advantage in time limited circumstances. To clarify this matter, further experiments with moderate time limits could be performed, and additonal data gathered to trace and plot convergence behaviour.

2. This chapter used a collection of real-world and synthetic data sets. Since the number of real data sets was limited by the number of ServicePower's clients, this does not provide a representative picture of all possible scheduling scenarios or use cases.

Even with these caveats, the benefits of parallelizing the workload of the Quantum Annealing algorithm are clear. The logical next phase of development is to distribute this workload further by employing clustered or cloud computing. Such technologies present additional opportunities but are beyond the scope of the KTP and this work.

The benefits to the business of ServicePower are presently counted in the currency of intellectual property (a US patent award with others pending), and the intangible value of marketing and advertising opportunities which help raise the profile of the company as being at the leading edge of technological development. Once fully integrated, tested and deployed as an end-to-end product, it can be demonstrated to customers who will no doubt see the potential of their workforces being scheduled by a quantum annealing algorithm.

# Chapter 6

# Conclusions and Perspectives

The original aim of this research was to develop and enhance the QA metaheuristic enough to make it a viable, scalable substitute for SA in the problem domains of VRP and FSS. Several of the main difficulties which prevent QA from wider application were exposed, investigated and then addressed. The creation of the spin encoding schemes highlighted ways that different problem domains may be mapped so they can be tackled with QA. The work went on to show that the tuner of algorithms need not resort to best-guesses, art forms, or arduous manual schemes. Beyond this, models were presented, implemented and then proved to show how, in multicore environments, population-based optimizers can compete with trajectory methods on the basis of wall-clock time.

From the development of QACVRP plus a basic tuning pattern, to the enhanced tuning method of ESPT and performance gains of PQAVRP, it is clear that the overall aim has been achieved and in several respects, surpassed. In collaboration with a private business, the output solution quality and performance of QA under real-world conditions was demonstrated to exceed that of SA. Moreover, this collaboration resulted in a patent award and added a new variant of QA (with stochastic kinetic energy) to the metaheuristic canon.

## 6.1 Review of contributions

### 6.1.1 Contributions of Chapter 2 - *Quantum Annealing Algorithm for the Vehicle Routing Problem*

Much of the work in this chapter was presented at the IEEE International Conference on Systems, Man and Cybernetics in 2013 (speaker Alex Syrichas):

Alan Crispin and Alex Syrichas. Quantum annealing algorithm for vehicle scheduling. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, October 2013. doi: 10.1109/smc.2013.601. URL `https://doi.org/10.1109/smc.2013.601`

- **First application of QA to VRP** When the QA implementation presented in Chapter 2 is placed into the context of being the first revision of the algorithm, then the computational results are impressive. The wall-clock times are never more than ten times slower than SA (except in the case of B-n67-k10), and in the two largest instances is faster (M-n121-k7, F-n135-k7). This is notable given that QA here is processing all of the replicas (40+) on a single thread of execution, and it is not fully tuned (this could not happen until the research of Chapter 3). Yet it is still yielding identical solution quality to SA in a reasonable amount of time.

- **Encoding of VRP as spin matrix** The encoding of VRP as spins was a derivative of that used for TSP [38], containing extensions for depots. The matrix was stored in memory in highly optimized bit-packed form which allowed for up to 64 spin-on-spin logical operations to occur simultaneously. It was this arrangement that enabled the run time of QA to remain reasonable.

### 6.1.2 Contributions of Chapter 3 - *Tuning QA for the Vehicle Routing Problem*

Much of the work in this chapter was published in the journal *Computers & Operations Research* by Elsevier:

A. Syrichas and A. Crispin. Large-scale vehicle routing problems: Quantum annealing, tunings and results. *Computers & Operations Research*, 87:52–62, November 2017. doi: 10.1016/j.cor.

2017.05.014. URL `https://doi.org/10.1016/j.cor.2017.05.014`

- **ESPT Tuning method** The dynamic cost model [6] which described the run time behaviour of QACVRP provided the key insight which was needed to develop the ESPT method. The fitness cloud representation of the behaviour showed that the topology of the fitness landscapes (within classes of instances) was similar enough for the predicted temperature value to succumb to tuning by a scaling factor.

- **New best solutions in benchmarks** Although the competitive testing paradigm is non-productive [6] in research such as this, it should be pointed out that these new best solutions were generated as a consequence of studying the fitness landscape, problem domain and the QA metaheuristic. Acknowledging this achievement will hopefully encourage others to work to understand why their algorithms perform well.

### 6.1.3 Contributions of Chapter 4 - *Quantum Annealing Algorithm: Enhancements and Variations*

Much of the work in this chapter is pending publication in the journal *Operations Research Perspectives* by Elsevier:

A. Syrichas and A. Crispin. A parallelized quantum annealing algorithm for vehicle routing problems. *Operations Research Perspectives*, 2019. (Under review)

- **New parallelized algorithm for QA of VRP** After the issue of the difficulty of tuning QA was addressed, it was natural and desirable to enhance the runtime performance of the algorithm. If QA is to become a solid choice of metaheuristic to supplant SA, then all key objections to deploying it must be quashed. It should be noted that unlocking the performance of QA strictly requires a platform capable of hosting a large population of threads.

### 6.1.4 Contributions of Chapter 5 - *The Industrial Domain of Field Service Scheduling*

- **First practical application of QA in industry** To the best of our knowledge, this is the first time QA has been used in an commercial role. That it is able to solve complex scheduling problems as well as idealized academic benchmarks is a notable achievement.
- **Three new QA-based algorithms for FSS** The path to the completion of a new scalable optimizer for FSS is punctuated with three milestones - Quasi-Quantum Annealing, Full Quantum Annealing, and Parallelized Quantum Annealing. QQA was a vital first step proving that a population-based metaheuristic was a good fit for the domain. It was a proof of concept that was shown to also be a minimum viable product since it could optimize nearly as well as FQA.
- **Patent approved for spin encoding of FSS problems** The design for the spin encoding of a FSS problem was awarded a patent by the US Patent Office. This underscores the novelty of matrix arrangement whilst also accruing valuable intellectual property and marketing opportunities for ServicePower.

## 6.2 Future directions of research

As with most projects that are given a latitude and (seemingly) enough time for exploration, several directions remain uncharted. Most obviously, QA could be tried in other problem domains. However, this would be the least interesting direction as the supply of domains is a near-inexhaustible list, consisting of permutations in constraints and objectives. (There is also the risk of entrapment in the competitive testing paradigm). The following offers future directions with more fruitful and perhaps, practical outcomes:

- **Distributed, clustered, and cloud computing.** QA has been shown to benefit greatly when the workload is distributed to separate threads of execution. The natural extension to this arrangement is to employ a network of computers to share the load. Compute clusters and cloud platforms seem mature enough to yield success in this direction. In principle, it is possible to deploy a QA optimization process which has a population of thousands of replicas - a highly accurate PIMC method which might locate ground states (optimal solutions) with unerring consistency. Additionally, the limit on the size of problem instances would be raised because additional memory could be assigned upon demand. Of course, the key here would be to find a network which can communicate states between replicas quickly enough.

- **A deeper exploration of dynamic cost models and fitness landscape analysis.** Predicting the behaviour of metaheuristics upon any problem instance, is one of the holy grails of optimization research. Even approximate predictions can garner useful insights and benefits. The work in Chapter 3 suggested that for QA, affine transformations might yield more accurate parameter values - if so, then the fitness cloud might be improved by the addition of extra dimensions.

# Bibliography

[1] International Federation of Operational Research Societies. What is OR? `http://ifors.org/what-is-or/`. Accessed 15 June 2019.

[2] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[3] Eugene L Lawler, Jan Karel Lenstra, AHG Rinnooy Kan, David Bernard Shmoys, et al. *The traveling salesman problem: a guided tour of combinatorial optimization*, volume 3. Wiley New York, 1985.

[4] Merrill M. Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, February 1956. doi: 10.1287/opre.4.1.61. URL `https://doi.org/10.1287/opre.4.1.61`.

[5] J. Christopher Beck, Patrick Prosser, and Evgeny Selensky. On the reformulation of vehicle routing problems and scheduling problems. In *Lecture Notes in Computer Science*, pages 282–289. Springer Berlin Heidelberg, 2002. doi: 10.1007/3-540-45622-8_21. URL `https://doi.org/10.1007/3-540-45622-8_21`.

[6] Jean-Paul Watson. An introduction to fitness landscape analysis and cost models for local search. In *Handbook of Metaheuristics*, pages 599–623. Springer US, 2010. doi: 10.1007/978-1-4419-1665-5_20. URL `https://doi.org/10.1007/978-1-4419-1665-5_20`.

[7] Shen Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, December 1965. doi: 10.1002/j.1538-7305.1965.tb04146.x. URL `https://doi.org/10.1002/j.1538-7305.1965.tb04146.x`.

[8] Paul M. Thompson and Harilaos N. Psaraftis. Cyclic transfer algorithm for multivehicle routing and scheduling problems. *Operations Research*, 41(5):935–946, October 1993. doi: 10.1287/opre.41.5.935. URL `https://doi.org/10.1287/opre.41.5.935`.

[9] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, February 2005. doi: 10.1287/trsc.1030.0056. URL `https://doi.org/10.1287/trsc.1030.0056`.

[10] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science*, 39(1):119–139, February 2005. doi: 10.1287/trsc.1030.0057. URL `https://doi.org/10.1287/trsc.1030.0057`.

[11] Bruno-Laurent Garcia, Jean-Yves Potvin, and Jean-Marc Rousseau. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Computers & Operations Research*, 21(9):1025–1033, November 1994. doi: 10.1016/0305-0548(94)90073-6. URL `https://doi.org/10.1016/0305-0548(94)90073-6`.

[12] Fred Glover. Tabu search—part i. *ORSA Journal on Computing*, 1(3):190–206, August 1989. doi: 10.1287/ijoc.1.3.190. URL `https://doi.org/10.1287/ijoc.1.3.190`.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983. doi: 10.1126/science.220.4598.671. URL `https://doi.org/10.1126/science.220.4598.671`.

[14] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, January 1985. doi: 10.1007/bf00940812. URL `https://doi.org/10.1007/bf00940812`.

[15] N Metropolis, AW Rosenbluth, MN Rosenbluth, AH Teller, and E Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953. doi: 10.1063/1.1699114. URL `https://doi.org/10.1063/1.1699114`.

[16] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970. doi: 10.1093/biomet/57.1.97. URL `https://doi.org/10.1093/biomet/57.1.97`.

[17] R.A. Rutenbar. Simulated annealing algorithms: an overview. *IEEE Circuits and Devices Magazine*, 5(1):19–26, January 1989. doi: 10.1109/101.17235. URL https://doi.org/10.1109/101.17235.

[18] Harold M. Hastings and Stefan Waner. Low dissipation computing in biological systems. *Biosystems*, 17(3):241–244, January 1985. doi: 10.1016/0303-2647(85)90078-4. URL https://doi.org/10.1016/0303-2647(85)90078-4.

[19] Z. Li and H. A. Scheraga. Monte carlo-minimization approach to the multiple-minima problem in protein folding. *Proceedings of the National Academy of Sciences*, 84(19):6611–6615, October 1987. doi: 10.1073/pnas.84.19.6611. URL https://doi.org/10.1073/pnas.84.19.6611.

[20] K. Bryan, P. Cunningham, and N. Bolshakova. Application of simulated annealing to the biclustering of gene expression data. *IEEE Transactions on Information Technology in Biomedicine*, 10(3):519–525, July 2006. doi: 10.1109/titb.2006.872073. URL https://doi.org/10.1109/titb.2006.872073.

[21] J. Brooke. Quantum annealing of a disordered magnet. *Science*, 284(5415):779–781, April 1999. doi: 10.1126/science.284.5415.779. URL https://doi.org/10.1126/science.284.5415.779.

[22] G. E. Santoro. Theory of quantum annealing of an ising spin glass. *Science*, 295(5564): 2427–2430, March 2002. doi: 10.1126/science.1068774. URL https://doi.org/10.1126/science.1068774.

[23] Demian A. Battaglia, Giuseppe E. Santoro, and Erio Tosatti. Optimization by quantum annealing: Lessons from hard satisfiability problems. *Physical Review E*, 71(6), June 2005. doi: 10.1103/physreve.71.066707. URL https://doi.org/10.1103/physreve.71.066707.

[24] O Titiloye and A Crispin. Quantum annealing of the graph coloring problem. *Discrete Optimization*, 8(2):376–384, May 2011. doi: 10.1016/j.disopt.2010.12.001. URL https://doi.org/10.1016/j.disopt.2010.12.001.

[25] Olawale Titiloye and Alan Crispin. Parameter tuning patterns for random graph coloring with quantum annealing. *PLoS ONE*, 7(11):e50060, November 2012. doi: 10.1371/journal.pone.0050060. URL `https://doi.org/10.1371/journal.pone.0050060`.

[26] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Mathematical Programming*, 34(1):111–124, January 1986. doi: 10.1007/bf01582166. URL `https://doi.org/10.1007/bf01582166`.

[27] Hime Aguiar e Oliveira Junior, Lester Ingber, Antonio Petraglia, Mariane Rembold Petraglia, and Maria Augusta Soares Machado. Adaptive simulated annealing. In *Intelligent Systems Reference Library*, pages 33–62. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-27479-4_4. URL `https://doi.org/10.1007/978-3-642-27479-4_4`.

[28] Matthew Crossley, Andy Nisbet, and Martyn Amos. Quantifying the impact of parameter tuning on nature-inspired algorithms. In *Advances in Artificial Life, ECAL 2013*. MIT Press, September 2013. doi: 10.7551/978-0-262-31709-2-ch138. URL `https://doi.org/10.7551/978-0-262-31709-2-ch138`.

[29] Oden Maron and Andrew W. Moore. *Artificial Intelligence Review*, 11(1/5):193–225, 1997. doi: 10.1023/a:1006556606079. URL `https://doi.org/10.1023/a:1006556606079`.

[30] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997. doi: 10.1109/4235.585893. URL `https://doi.org/10.1109/4235.585893`.

[31] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 11–18. Morgan Kaufmann Publishers Inc., 2002.

[32] Eliezer Yudkowsky. Optimization. `https://www.lesswrong.com/posts/D7EcMhL26zFNbJ3ED/optimization`. Accessed 15 June 2019.

[33] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, August 1964. doi: 10.1287/opre.12.4.568. URL `https://doi.org/10.1287/opre.12.4.568`.

[34] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, January 1986. doi: 10.1016/0305-0548(86)90048-1. URL `https://doi.org/10.1016/0305-0548(86)90048-1`.

[35] Chris Groër, Bruce Golden, and Edward Wasil. A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, 2(2):79–101, April 2010. doi: 10.1007/s12532-010-0013-5. URL `https://doi.org/10.1007/s12532-010-0013-5`.

[36] Christian Blum, Andrea Roli, and Enrique Alba. An introduction to metaheuristic techniques. In *Parallel Metaheuristics*, pages 1–42. John Wiley & Sons, Inc., September 2005. doi: 10.1002/0471739383.ch1. URL `https://doi.org/10.1002/0471739383.ch1`.

[37] Edmund K. Burke and Yuri Bykov. The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70–78, April 2017. doi: 10.1016/j.ejor.2016.07.012. URL `https://doi.org/10.1016/j.ejor.2016.07.012`.

[38] Roman Martoňák, Giuseppe E. Santoro, and Erio Tosatti. Quantum annealing of the traveling-salesman problem. *Physical Review E*, 70(5), November 2004. doi: 10.1103/physreve.70.057701. URL `https://doi.org/10.1103/physreve.70.057701`.

[39] A. Syrichas and A. Crispin. A parallelized quantum annealing algorithm for vehicle routing problems. *Operations Research Perspectives*, 2019. (Under review).

[40] A. Syrichas and A. Crispin. Encoding of a schedule into a binary structure, 2017. US Patent 9,841,990.

[41] Alan Crispin and Alex Syrichas. Quantum annealing algorithm for vehicle scheduling. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, October 2013. doi: 10.1109/smc.2013.601. URL `https://doi.org/10.1109/smc.2013.601`.

[42] A. Syrichas and A. Crispin. Large-scale vehicle routing problems: Quantum annealing, tunings and results. *Computers & Operations Research*, 87:52–62, November 2017. doi: 10.1016/j.cor.2017.05.014. URL `https://doi.org/10.1016/j.cor.2017.05.014`.

[43] Manchester Metropolitan University Press Release. Quantum physics-based algorithm helps companies better manage mobile workforces: University's patented code powers field service management software, 2018. URL `http://www2.mmu.ac.uk/news-and-events/news/story/7359`. Accessed 23 June 2019.

[44] Manchester Metropolitan University Press Release. Servicepower case study: State of the art software helps industry improve vehicle logistics by up to 10%, 2017. URL `https://www2.mmu.ac.uk/business/success-stories/detail/service-power.php`. Accessed 23 June 2019.

[45] Electronic Times: Software Simulations Now. Quantum computing uses standard hardware, 2017. URL `https://www.eetimes.com/document.asp?doc_id=1331768`. Accessed 23 June 2019.

[46] Field Service News. Is quantum computing the future of customer experience and service optimisation?, 2017. URL `http://fieldservicenews.com/quantum-computing-future-customer-experience-service-optimisation`. Accessed 23 June 2019.

[47] Manchester Metropolitan University Press Release. Quantum computing solves logistics conundrum. algorithm helps lorries deliver goods efficiently, 2016. URL `http://www.mmu.ac.uk/news/news-items/4204`. Accessed 23 June 2019.

[48] ServicePower Inc. David and goliath: Redefining field service with quantum annealing, 2016. URL `https://www.servicepower.com/understanding-quantum-annealing-awareness-lp`. Accessed 23 June 2019.

[49] World News. Quantum computing solves logistics conundrum, 2016. URL `http://article.wn.com/view/2016/03/10/Quantum_computing_solves_logistics_conundrum_Manchester_Metr`. Accessed 23 June 2019.

[50] ServicePower Inc. Servicepower applies for various quantum annealing patents, 2015. URL `https://www.servicepower.com/blog/servicepower-applies-for-various-quantum-annealing-patents`. Accessed 23 June 2019.

[51] G Rose and W Macready. An introduction to quantum annealing. *D-Wave Systems*, 2007. URL `http://dwave.files.wordpress.com/2007/08/20070810_d-wave_quantum_annealing.pdf`. Accessed 21 May 2012.

[52] Arnab Das and Bikas K. Chakrabarti. Colloquium: Quantum annealing and analog quantum computation. *Reviews of Modern Physics*, 80(3):1061–1081, September 2008. doi: 10.1103/revmodphys.80.1061. URL `https://doi.org/10.1103/revmodphys.80.1061`.

[53] T Ralphs. Branch cut and price resource, vehicle routing data sets. `http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/index.htm`. Accessed: 30 January 2015.

[54] Áslaug Sóley Bjarnadóttir. Solving the vehicle routing problem with genetic algorithms. Master's thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2004.

[55] István Burgulya. A memetic algorithm for the capacitated vehicle routing problem. *Acta Agraria Kaposváriensis*, 12(2):59–69, 2008.

[56] Ajith Abraham and Crina Grosan. Engineering evolutionary intelligent systems: Methodologies, architectures and reviews. In *Engineering Evolutionary Intelligent Systems*, pages 1–22. Springer Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-75396-4_1. URL `https://doi.org/10.1007/978-3-540-75396-4_1`.

[57] Enda Ridge and Daniel Kudenko. Tuning an algorithm using design of experiments. In *Experimental Methods for the Analysis of Optimization Algorithms*, pages 265–286. Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-02538-9_11. URL `https://doi.org/10.1007/978-3-642-02538-9_11`.

[58] Hui Nie, Bo Liu, Pumo Xie, Zhenbing Liu, and Huihua Yang. A cuckoo search algorithm for scheduling multiskilled workforce. *Journal of Networks*, 9(5), May 2014. doi: 10.4304/jnw.9.5.1346-1353. URL `https://doi.org/10.4304/jnw.9.5.1346-1353`.

[59] P Augerat, JM Belenguer, E Benavent, A Corberán, D Naddef, and G Rinaldi. *Computational results with a branch and cut code for the capacitated vehicle routing problem*. IMAG, 1995.

[60] P Collard, S Vérel, and M Clergue. Local search heuristics: Fitness cloud versus fitness landscape. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI 2004)*, pages 973–974.

[61] Leonardo Vanneschi, Marco Tomassini, Philippe Collard, and Sébastien Vérel. Negative slope coefficient: A measure to characterize genetic programming fitness landscapes. In *Lecture Notes in Computer Science*, pages 178–189. Springer Berlin Heidelberg, 2006. doi: 10.1007/11729976_16. URL `https://doi.org/10.1007/11729976_16`.

[62] É. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8): 661–673, December 1993. doi: 10.1002/net.3230230804. URL `https://doi.org/10.1002/net.3230230804`.

[63] David Mester and Olli Bräysy. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, 32(6):1593–1614, June 2005. doi: 10.1016/j.cor.2003.11.017. URL `https://doi.org/10.1016/j.cor.2003.11.017`.

[64] Roberto De Franceschi, Matteo Fischetti, and Paolo Toth. A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105(2-3):471–499, November 2005. doi: 10.1007/s10107-005-0662-8. URL `https://doi.org/10.1007/s10107-005-0662-8`.

[65] Enrique Alba and Bernabé Dorronsoro. Computing nine new best-so-far solutions for capacitated VRP with a cellular genetic algorithm. *Information Processing Letters*, 98(6): 225–230, June 2006. doi: 10.1016/j.ipl.2006.02.006. URL `https://doi.org/10.1016/j.ipl.2006.02.006`.

[66] Marcus Poggi and Eduardo Uchoa. Chapter 3: New exact algorithms for the capacitated vehicle routing problem. In *Vehicle Routing*, pages 59–86. Society for Industrial and Applied Mathematics, November 2014. doi: 10.1137/1.9781611973594.ch3. URL `https://doi.org/10.1137/1.9781611973594.ch3`.

[67] Wanfeng Liu and Xia Li. A problem-reduction evolutionary algorithm for solving the capacitated vehicle routing problem. *Mathematical Problems in Engineering*, 2015:1–11, 2015. doi: 10.1155/2015/165476. URL `https://doi.org/10.1155/2015/165476`.

[68] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20(3):309–318, September 1969. doi: 10.1057/jors.1969.75. URL `https://doi.org/10.1057/jors.1969.75`.

[69] Billy E Gillett and Jerry G Johnson. Multi-terminal vehicle-dispatch algorithm. *Omega*, 4 (6):711–718, January 1976. doi: 10.1016/0305-0483(76)90097-9. URL `https://doi.org/10.1016/0305-0483(76)90097-9`.

[70] D Ratliff, N Christofides, A Mingozzi, P Toth, and C Sandi. Combinatorial optimization, 1981. URL `http://www.jstor.org/stable/25060157`.

[71] Bruce L. Golden, Edward A. Wasil, James P. Kelly, and I-Ming Chao. The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In *Fleet Management and Logistics*, pages 33–56. Springer US, 1998. doi: 10.1007/978-1-4615-5755-5_2. URL `https://doi.org/10.1007/978-1-4615-5755-5_2`.

[72] C.D. Tarantilis and C.T. Kiranoudis. Boneroute: An adaptive memory-based method for effective fleet management. *Annals of Operations Research*, 115(1/4):227–241, 2002. doi: 10.1023/a:1021157406318. URL `https://doi.org/10.1023/a:1021157406318`.

[73] F Li, B Golden, and E Wasil. Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, 32(5):1165–1179, May 2005. doi: 10.1016/j.cor.2003.10.002. URL `https://doi.org/10.1016/j.cor.2003.10.002`.

[74] KH Han, JH Kim, et al. Analysis of quantum-inspired evolutionary algorithm. In *Proceedings of the 2001 International Conference on Artificial Intelligence*, pages 727–730. Citeseer, 2001.

[75] HQ Zheng, YQ Zhou, and QF Luo. A hybrid cuckoo search algorithm-GRASP for vehicle routing problem. *Journal of Convergence Information Technology*, 8(3):821–828, Febru-

ary 2013. doi: 10.4156/jcit.vol8.issue3.97. URL https://doi.org/10.4156/jcit.vol8.issue3.97.

[76] Innovate UK. Knowledge transfer partnerships. http://ktp.innovateuk.org/. Accessed 21 May 2019.

[77] UK Government. Innovate uk. https://www.gov.uk/government/organisations/innovate-uk. Accessed 21 May 2019.

[78] Dyan L. Haugen and Arthur V. Hill. Scheduling to improve field service quality. *Decision Sciences*, 30(3):783–804, June 1999. doi: 10.1111/j.1540-5915.1999.tb00906.x. URL https://doi.org/10.1111/j.1540-5915.1999.tb00906.x.

[79] Victor Pillac, Christelle Guéret, and Andrés Medaglia. On the technician routing and scheduling problem. In *Metaheuristics International Conference (MIC) 2011*, pages S2–40, 2011.

[80] S. Xavier de Souza, J.A.K. Suykens, J. Vandewalle, and D. Bolle. Coupled simulated annealing. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40 (2):320–335, April 2010. doi: 10.1109/tsmcb.2009.2020435. URL https://doi.org/10.1109/tsmcb.2009.2020435.

[81] George Ruppeiner, Jacob Mørch Pedersen, and Peter Salamon. Ensemble approach to simulated annealing. *Journal de Physique I*, 1(4):455–470, April 1991. doi: 10.1051/jp1:1991146. URL https://doi.org/10.1051/jp1:1991146.

# Appendices

# A Programming: Platforms and tools

Versions of the C++ language from C++11 onwards have standardized support for multithreaded programming. Usage of the Thread Support Library eliminates the worry of cross-platform portability (previously platform-specific APIs had to be compiled or linked against to take advantage of multithreaded environments). The latest GNU C++ compiler is used as it has repeatedly proven itself ISO compliant (currently ISO/IEC 14882:2014 and ISO/IEC TS 19217:2015 experimentally).

A Linux distribution hosted upon a multicore workstation is the ideal environment for the development and testing of, and experimention upon, multithreaded QA algorithms.

## A.1 Computing Platform 1

**Operating System**: Ubuntu 11.04 (natty) Kernel 2.6.38-11-generic

**Computing system**: Generic desktop PC

**Processor**: Intel(R) Pentium(R) Dual-Core CPU E5800 @ 3.20GHz

**RAM**: 4Gb

## A.2 Computing platform 2

**Operating System**: Debian 3.16.51-3+deb8u1 (2018-01-08), release: 3.16.0-5-amd64

**Computing system**: Hewlett Packard HP Z640 Workstation

**Processor**: Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz

**RAM**: 32Gb

## A.3 Software environment

The software development environment was as follows:

**Compiler/Linker**: GNU C (gcc) 4.9.2

**IDE**: Qt Creator 3.3.1

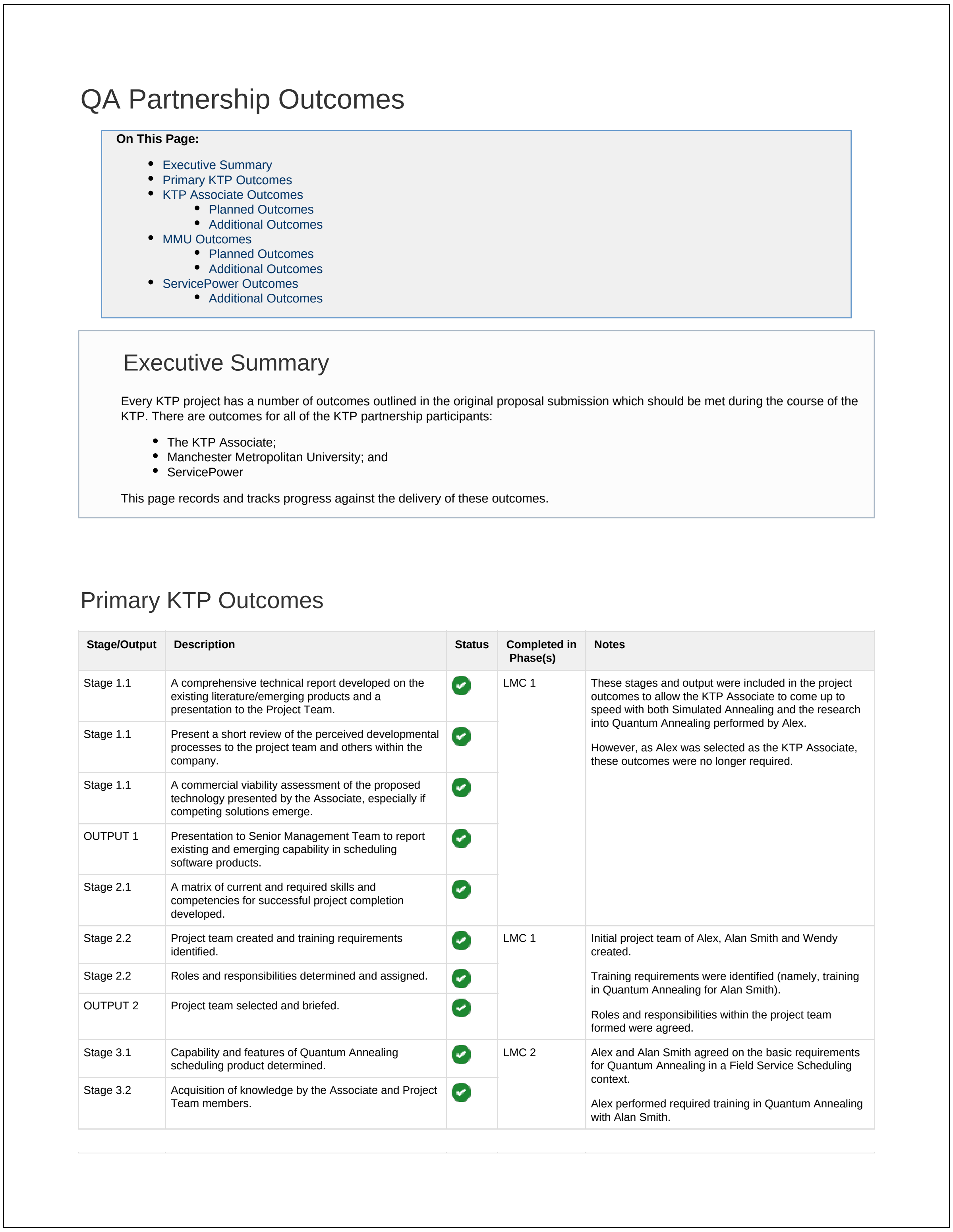
**Platform framework**: Qt 5.4.1

# B Knowledge Transfer Partnership Outcomes

The contents of this appendix gives an overview of the plan and a brief glimpse of the outputs from the completed KTP project to apply quantum annealing in an industrial setting.

## B.1 Project plan with outputs

The following is a reproduction of the Knowledge Transfer Programme (KTP) project plan. The milestones are divided into 7 stages which were distributed over 24 months. One or more outputs were aligned to each stage, and after each 4 month interval these and other progress points were presented and discussed at the Local Management Committee (LMC) meetings. Delivery of stages are indicated with check marks. As can be seen, Quantum Annealing for Field Service Scheduling Problems was delivered at stage 5.1 and experiments conducted at 5.3. Owing to the lengthy application process, output 8, consisting of four patent applications (one successful) was completed early and out of sequence.



## QA Partnership Outcomes

**On This Page:**

### Executive Summary

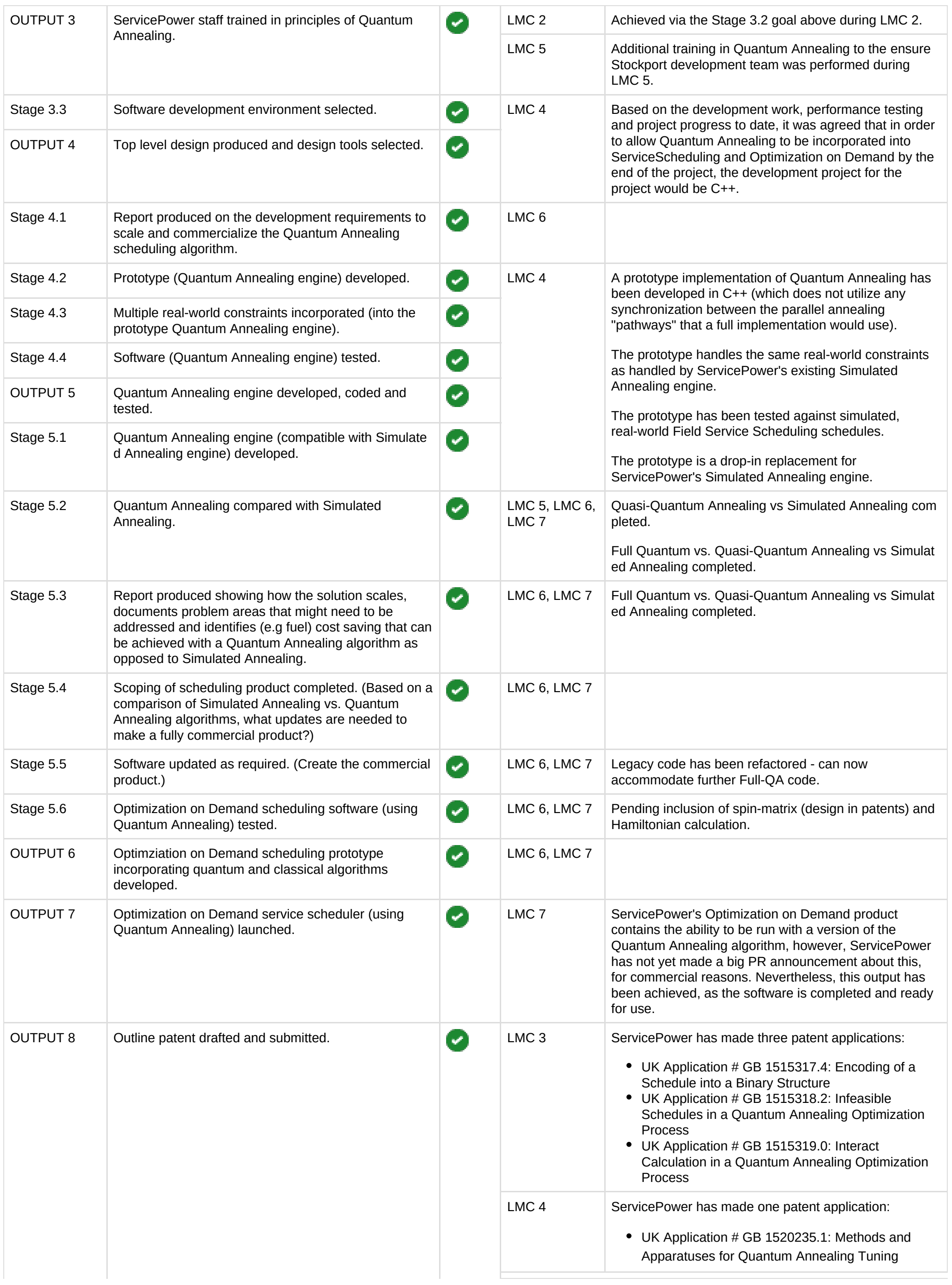
Every KTP project has a number of outcomes outlined in the original proposal submission which should be met during the course of the KTP. There are outcomes for all of the KTP partnership participants:

- The KTP Associate;
- Manchester Metropolitan University; and
- ServicePower

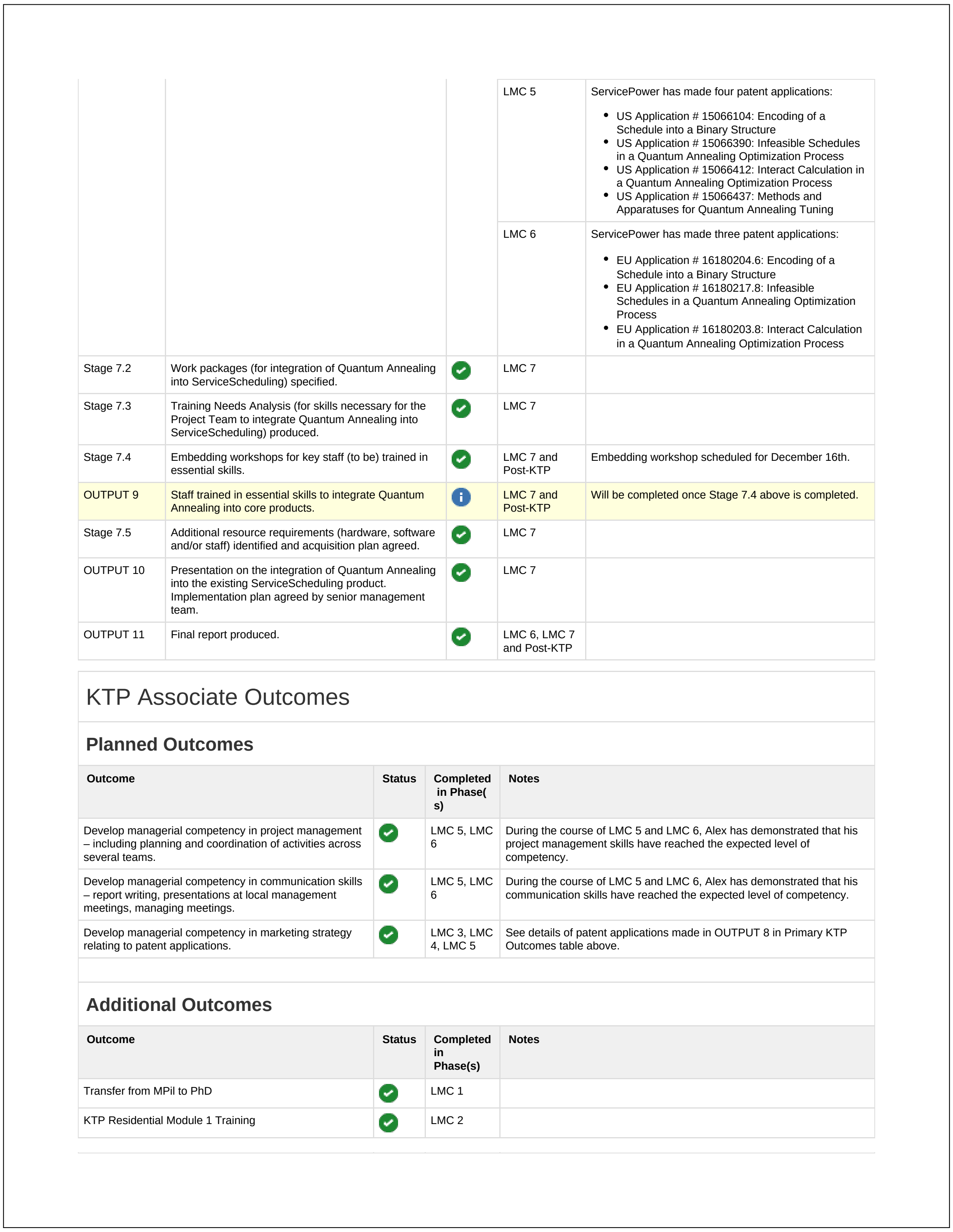
This page records and tracks progress against the delivery of these outcomes.

### Primary KTP Outcomes

| Stage/Output | Description | Status | Completed in Phase(s) | Notes |
|---|---|---|---|---|
| Stage 1.1 | A comprehensive technical report developed on the existing literature/emerging products and a presentation to the Project Team. | ✓ | LMC 1 | These stages and output were included in the project outcomes to allow the KTP Associate to come up to speed with both Simulated Annealing and the research into Quantum Annealing performed by Alex.<br><br>However, as Alex was selected as the KTP Associate, these outcomes were no longer required. |
| Stage 1.1 | Present a short review of the perceived developmental processes to the project team and others within the company. | ✓ | | |
| Stage 1.1 | A commercial viability assessment of the proposed technology presented by the Associate, especially if competing solutions emerge. | ✓ | | |
| OUTPUT 1 | Presentation to Senior Management Team to report existing and emerging capability in scheduling software products. | ✓ | | |
| Stage 2.1 | A matrix of current and required skills and competencies for successful project completion developed. | ✓ | | |
| Stage 2.2 | Project team created and training requirements identified. | ✓ | LMC 1 | Initial project team of Alex, Alan Smith and Wendy created.<br><br>Training requirements were identified (namely, training in Quantum Annealing for Alan Smith).<br><br>Roles and responsibilities within the project team formed were agreed. |
| Stage 2.2 | Roles and responsibilities determined and assigned. | ✓ | | |
| OUTPUT 2 | Project team selected and briefed. | ✓ | | |
| Stage 3.1 | Capability and features of Quantum Annealing scheduling product determined. | ✓ | LMC 2 | Alex and Alan Smith agreed on the basic requirements for Quantum Annealing in a Field Service Scheduling context.<br><br>Alex performed required training in Quantum Annealing with Alan Smith. |
| Stage 3.2 | Acquisition of knowledge by the Associate and Project Team members. | ✓ | | |



| | | | | |
|---|---|---|---|---|
| OUTPUT 3 | ServicePower staff trained in principles of Quantum Annealing. | ✓ | LMC 2 | Achieved via the Stage 3.2 goal above during LMC 2. |
| | | | LMC 5 | Additional training in Quantum Annealing to the ensure Stockport development team was performed during LMC 5. |
| Stage 3.3 | Software development environment selected. | ✓ | LMC 4 | Based on the development work, performance testing and project progress to date, it was agreed that in order to allow Quantum Annealing to be incorporated into ServiceScheduling and Optimization on Demand by the end of the project, the development project for the project would be C++. |
| OUTPUT 4 | Top level design produced and design tools selected. | ✓ | | |
| Stage 4.1 | Report produced on the development requirements to scale and commercialize the Quantum Annealing scheduling algorithm. | ✓ | LMC 6 | |
| Stage 4.2 | Prototype (Quantum Annealing engine) developed. | ✓ | LMC 4 | A prototype implementation of Quantum Annealing has been developed in C++ (which does not utilize any synchronization between the parallel annealing "pathways" that a full implementation would use).<br><br>The prototype handles the same real-world constraints as handled by ServicePower's existing Simulated Annealing engine.<br><br>The prototype has been tested against simulated, real-world Field Service Scheduling schedules.<br><br>The prototype is a drop-in replacement for ServicePower's Simulated Annealing engine. |
| Stage 4.3 | Multiple real-world constraints incorporated (into the prototype Quantum Annealing engine). | ✓ | | |
| Stage 4.4 | Software (Quantum Annealing engine) tested. | ✓ | | |
| OUTPUT 5 | Quantum Annealing engine developed, coded and tested. | ✓ | | |
| Stage 5.1 | Quantum Annealing engine (compatible with Simulate d Annealing engine) developed. | ✓ | | |
| Stage 5.2 | Quantum Annealing compared with Simulated Annealing. | ✓ | LMC 5, LMC 6, LMC 7 | Quasi-Quantum Annealing vs Simulated Annealing com pleted.<br><br>Full Quantum vs. Quasi-Quantum Annealing vs Simulat ed Annealing completed. |
| Stage 5.3 | Report produced showing how the solution scales, documents problem areas that might need to be addressed and identifies (e.g fuel) cost saving that can be achieved with a Quantum Annealing algorithm as opposed to Simulated Annealing. | ✓ | LMC 6, LMC 7 | Full Quantum vs. Quasi-Quantum Annealing vs Simulat ed Annealing completed. |
| Stage 5.4 | Scoping of scheduling product completed. (Based on a comparison of Simulated Annealing vs. Quantum Annealing algorithms, what updates are needed to make a fully commercial product?) | ✓ | LMC 6, LMC 7 | |
| Stage 5.5 | Software updated as required. (Create the commercial product.) | ✓ | LMC 6, LMC 7 | Legacy code has been refactored - can now accommodate further Full-QA code. |
| Stage 5.6 | Optimization on Demand scheduling software (using Quantum Annealing) tested. | ✓ | LMC 6, LMC 7 | Pending inclusion of spin-matrix (design in patents) and Hamiltonian calculation. |
| OUTPUT 6 | Optimziation on Demand scheduling prototype incorporating quantum and classical algorithms developed. | ✓ | LMC 6, LMC 7 | |
| OUTPUT 7 | Optimization on Demand service scheduler (using Quantum Annealing) launched. | ✓ | LMC 7 | ServicePower's Optimization on Demand product contains the ability to be run with a version of the Quantum Annealing algorithm, however, ServicePower has not yet made a big PR announcement about this, for commercial reasons. Nevertheless, this output has been achieved, as the software is completed and ready for use. |
| OUTPUT 8 | Outline patent drafted and submitted. | ✓ | LMC 3 | ServicePower has made three patent applications:<br>• UK Application # GB 1515317.4: Encoding of a Schedule into a Binary Structure<br>• UK Application # GB 1515318.2: Infeasible Schedules in a Quantum Annealing Optimization Process<br>• UK Application # GB 1515319.0: Interact Calculation in a Quantum Annealing Optimization Process |
| | | | LMC 4 | ServicePower has made one patent application:<br>• UK Application # GB 1520235.1: Methods and Apparatuses for Quantum Annealing Tuning |



| | | | | |
|---|---|---|---|---|
| | | | LMC 5 | ServicePower has made four patent applications:<br>• US Application # 15066104: Encoding of a Schedule into a Binary Structure<br>• US Application # 15066390: Infeasible Schedules in a Quantum Annealing Optimization Process<br>• US Application # 15066412: Interact Calculation in a Quantum Annealing Optimization Process<br>• US Application # 15066437: Methods and Apparatuses for Quantum Annealing Tuning |
| | | | LMC 6 | ServicePower has made three patent applications:<br>• EU Application # 16180204.6: Encoding of a Schedule into a Binary Structure<br>• EU Application # 16180217.8: Infeasible Schedules in a Quantum Annealing Optimization Process<br>• EU Application # 16180203.8: Interact Calculation in a Quantum Annealing Optimization Process |
| Stage 7.2 | Work packages (for integration of Quantum Annealing into ServiceScheduling) specified. | ✓ | LMC 7 | |
| Stage 7.3 | Training Needs Analysis (for skills necessary for the Project Team to integrate Quantum Annealing into ServiceScheduling) produced. | ✓ | LMC 7 | |
| Stage 7.4 | Embedding workshops for key staff (to be) trained in essential skills. | ✓ | LMC 7 and Post-KTP | Embedding workshop scheduled for December 16th. |
| OUTPUT 9 | Staff trained in essential skills to integrate Quantum Annealing into core products. | ℹ | LMC 7 and Post-KTP | Will be completed once Stage 7.4 above is completed. |
| Stage 7.5 | Additional resource requirements (hardware, software and/or staff) identified and acquisition plan agreed. | ✓ | LMC 7 | |
| OUTPUT 10 | Presentation on the integration of Quantum Annealing into the existing ServiceScheduling product. Implementation plan agreed by senior management team. | ✓ | LMC 7 | |
| OUTPUT 11 | Final report produced. | ✓ | LMC 6, LMC 7 and Post-KTP | |

## KTP Associate Outcomes

### Planned Outcomes

| Outcome | Status | Completed in Phase(s) | Notes |
|---|---|---|---|
| Develop managerial competency in project management – including planning and coordination of activities across several teams. | ✓ | LMC 5, LMC 6 | During the course of LMC 5 and LMC 6, Alex has demonstrated that his project management skills have reached the expected level of competency. |
| Develop managerial competency in communication skills – report writing, presentations at local management meetings, managing meetings. | ✓ | LMC 5, LMC 6 | During the course of LMC 5 and LMC 6, Alex has demonstrated that his communication skills have reached the expected level of competency. |
| Develop managerial competency in marketing strategy relating to patent applications. | ✓ | LMC 3, LMC 4, LMC 5 | See details of patent applications made in OUTPUT 8 in Primary KTP Outcomes table above. |

### Additional Outcomes

| Outcome | Status | Completed in Phase(s) | Notes |
|---|---|---|---|
| Transfer from MPil to PhD | ✓ | LMC 1 | |
| KTP Residential Module 1 Training | ✓ | LMC 2 | |

| | | | |
|---|---|---|---|
| Mentoring Experience | ✓ | LMC 2, LMC 4 | Alex has gained valuable experience in mentoring others, through his work mentoring Amy in her PhD. |
| KTP Residential Module 2 Training | ✓ | LMC 3 | Alex learned more about the marketing and finance background aspects of business.<br><br>Alex was part of a team that beat all previous records during the Business Simulation task (using SIMCO software). |
| Patent Process Exposure | ✓ | LMC 3 | Alex has had exposure to the process of applying for patents, which has also helped his understanding of the mathematics of Quantum Annealing, leading to a more robust implementation of the algorithm in code. |
| Academic Exposure | ✓ | LMC 3 | The patent application process has raised Alex's profile at MMU with the Head of School, Kieth Miller. |
| Training | ✓ | LMC 4 | Alex has had exposure to new visualization techniques through MMU.<br><br>Alex has had the opportunity to be exposed to project management through mentoring with ServicePower. |
| Cloud Computing Experience | ✓ | LMC 5 | Alex has had the opportunity to use cloud-computing systems provided by ServicePower as part of running his Quantum Annealing experiments. |
| Employment from ServicePower post-KTP | ✓ | LMC 6 | Alex has been offered and accepted a contract to work for ServicePower following on from the completion of the KTP project. |

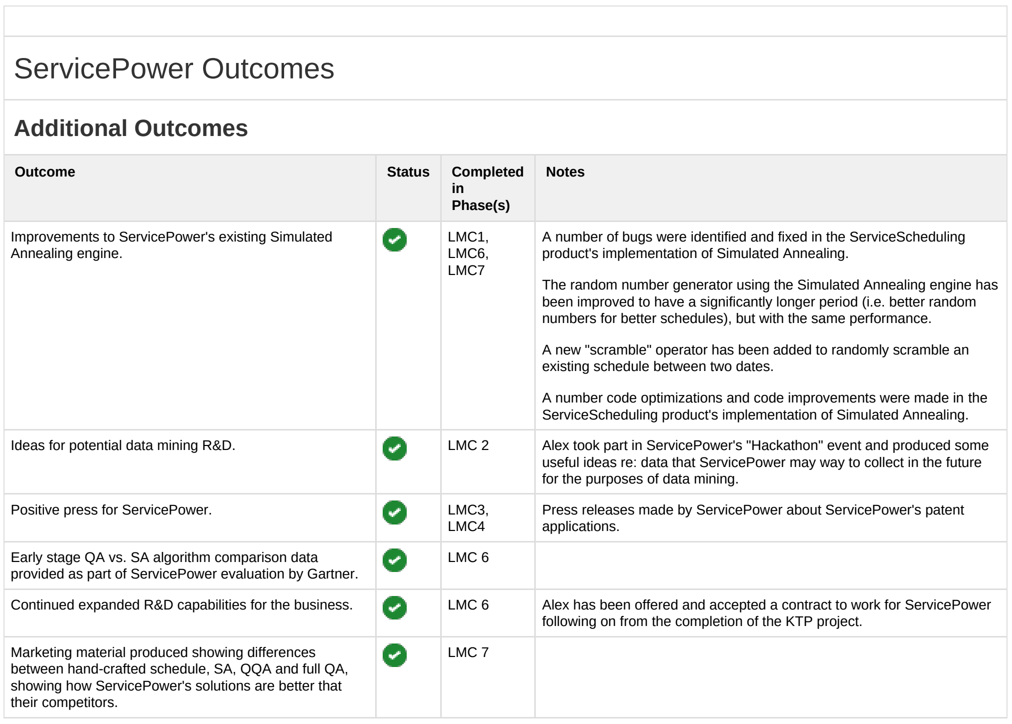
## MMU Outcomes

### Planned Outcomes

| Outcome | Status | Completed in Phase(s) | Notes |
|---|---|---|---|
| One conference presentation (e.g. IEEE or KES) and one journal publication (e.g. IEEE), depending on company requirements for confidentiality and patent application restrictions. | ✓ | LMC 5, LMC 6, LMC 7 an d Post-KTP | One journal publication submission has been made (in LMC 5) and accepted subject to some requested changes (in LMC 7). Changes and additional experiments to support the paper were made in LMC 7, with re-submission planned shortly after the KTP.<br><br>Due to outcomes obtained, instead of a conference presentation, a second journal paper has been agreed on, and has been worked on during LMC 6 and LMC 7. Submission is expected shortly after the KTP. |
| Patent developed in applying Quantum Annealing to field service scheduling. | ✓ | LMC 3, LMC 4, LMC 5 | See details of patent applications made in OUTPUT 8 in Primary KTP Outcomes table above. |
| Two undergraduate (timetable scheduling, traveling sales man) and postgraduate student projects (vehicle routing, quantum computation). | ✓ | Whole KTP | Dr. Crispin has run numerous undergraduate and postgraduate student projects in the related fields of scheduling and optimization problems. |
| Student placement and student visits. | ✓ | Whole KTP | ServicePower has hosted two placement students over the course of the KTP. |
| Industrially sponsored Ph.D. studentship (Scheduling Algorithms). | ✓ | LMC 1 | Funding for Alex's PhD has been obtained. |

### Additional Outcomes

| Outcome | Status | Completed in Phase(s) | Notes |
|---|---|---|---|
| Exposure to industrial software development practices. | ✓ | LMC 4 | Dr. Cripsin has had the opportunity to be exposed to industrial software development practices through collaboration with ServicePower. |
| Positive press for MMU. | ✓ | LMC 5 | Press releases made by MMU about KTP progress to date. |
| Potential Impact Case Study for mock REF (Research Excellent Framework) exercise. | ✓ | LMC 6 | Potential for including work on KTP with ServicePower in REF exercises. |
| KTP project included in an umbrella nomination for MMU KE (Knowledge Exchange) award. | ✓ | LMC 6 | Nomination received a "Highly Commended" result. |



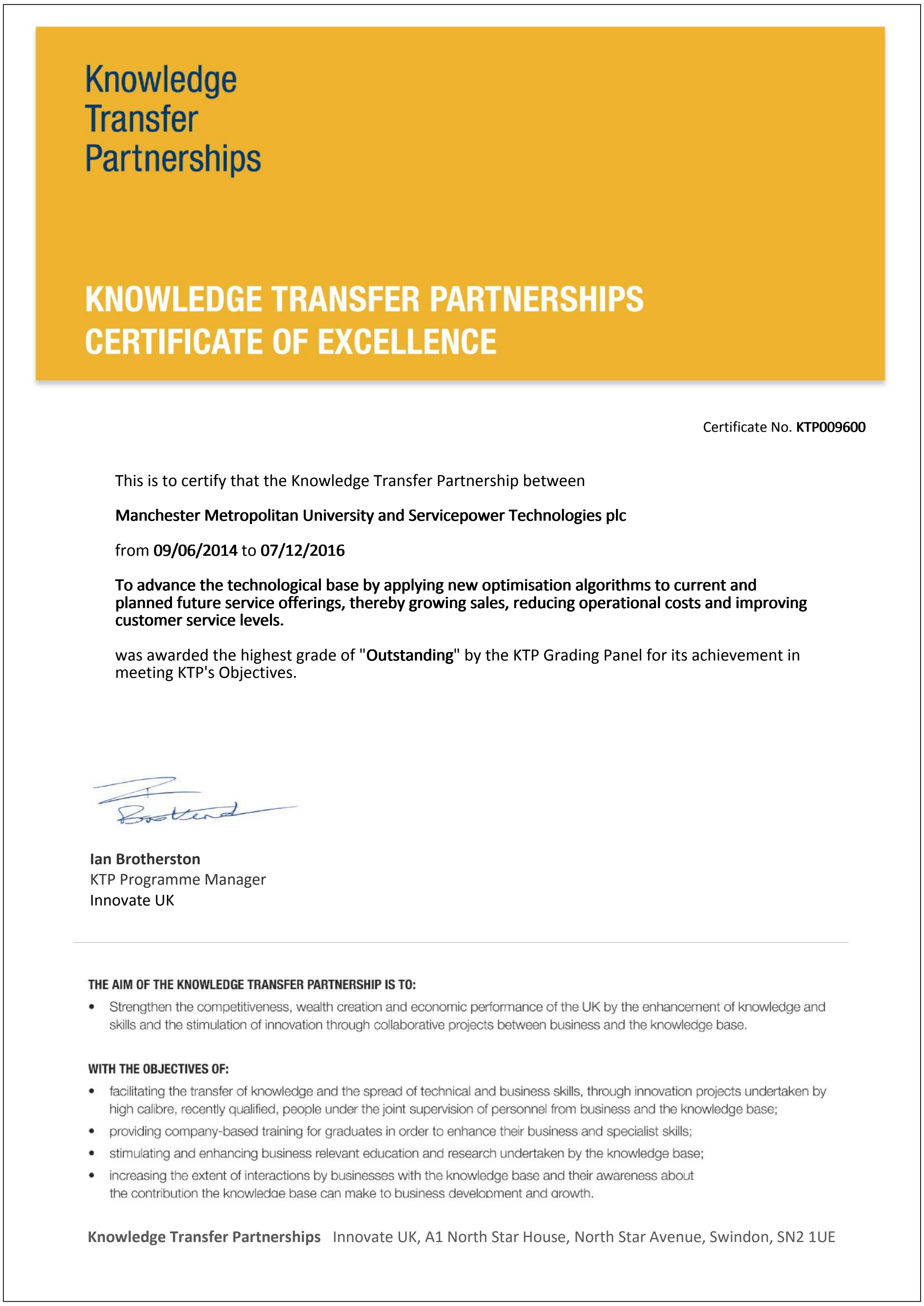
## ServicePower Outcomes

### Additional Outcomes

| Outcome | Status | Completed in Phase(s) | Notes |
|---|---|---|---|
| Improvements to ServicePower's existing Simulated Annealing engine. | ✓ | LMC1, LMC6, LMC7 | A number of bugs were identified and fixed in the ServiceScheduling product's implementation of Simulated Annealing.<br><br>The random number generator using the Simulated Annealing engine has been improved to have a significantly longer period (i.e. better random numbers for better schedules), but with the same performance.<br><br>A new "scramble" operator has been added to randomly scramble an existing schedule between two dates.<br><br>A number code optimizations and code improvements were made in the ServiceScheduling product's implementation of Simulated Annealing. |
| Ideas for potential data mining R&D. | ✓ | LMC 2 | Alex took part in ServicePower's "Hackathon" event and produced some useful ideas re: data that ServicePower may way to collect in the future for the purposes of data mining. |
| Positive press for ServicePower. | ✓ | LMC3, LMC4 | Press releases made by ServicePower about ServicePower's patent applications. |
| Early stage QA vs. SA algorithm comparison data provided as part of ServicePower evaluation by Gartner. | ✓ | LMC 6 | |
| Continued expanded R&D capabilities for the business. | ✓ | LMC 6 | Alex has been offered and accepted a contract to work for ServicePower following on from the completion of the KTP project. |
| Marketing material produced showing differences between hand-crafted schedule, SA, QQA and full QA, showing how ServicePower's solutions are better that their competitors. | ✓ | LMC 7 | |

## B.2 Certificate

The KTP project (KTP009600) was reviewed by Innovate UK and attained the award of "Outstanding". The certificate showing this award is reproduced below.



Knowledge
Transfer
Partnerships

KNOWLEDGE TRANSFER PARTNERSHIPS
CERTIFICATE OF EXCELLENCE

Certificate No. **KTP009600**

This is to certify that the Knowledge Transfer Partnership between

**Manchester Metropolitan University and Servicepower Technologies plc**

from **09/06/2014** to **07/12/2016**

**To advance the technological base by applying new optimisation algorithms to current and planned future service offerings, thereby growing sales, reducing operational costs and improving customer service levels.**

was awarded the highest grade of "**Outstanding**" by the KTP Grading Panel for its achievement in meeting KTP's Objectives.

[illegible]

**Ian Brotherston**
KTP Programme Manager
Innovate UK

**THE AIM OF THE KNOWLEDGE TRANSFER PARTNERSHIP IS TO:**

- Strengthen the competitiveness, wealth creation and economic performance of the UK by the enhancement of knowledge and skills and the stimulation of innovation through collaborative projects between business and the knowledge base.

**WITH THE OBJECTIVES OF:**

- facilitating the transfer of knowledge and the spread of technical and business skills, through innovation projects undertaken by high calibre, recently qualified, people under the joint supervision of personnel from business and the knowledge base;
- providing company-based training for graduates in order to enhance their business and specialist skills;
- stimulating and enhancing business relevant education and research undertaken by the knowledge base;
- increasing the extent of interactions by businesses with the knowledge base and their awareness about the contribution the knowledge base can make to business development and growth.

**Knowledge Transfer Partnerships** Innovate UK, A1 North Star House, North Star Avenue, Swindon, SN2 1UE

## B.3 Patent

The following is a three page excerpt of the awarded US patent, the production of which was a task highly prioritized by ServicePower:

A. Syrichas and A. Crispin. Encoding of a schedule into a binary structure, 2017. US Patent 9,841,990 It describes a method of storing a complex schedule as structured collection of bits, allowing a compact representation of a spin matrix which can be efficiently processed during the simulation of quantum tunneling in the annealing algorithms.

# ENCODING OF A SCHEDULE INTO A STRUCTURE

## TECHNICAL FIELD

The present invention relates to a method, apparatus and system for encoding of a schedule into a structure for use in a Quantum Annealing (QA) optimisation process. This is particularly useful for, but not limited to, the Field Services Scheduling (FSS) industry where complex schedules have to be optimised.

## BACKGROUND

When contemplating the problem of optimising a schedule (i.e. of identifying a schedule that is considered as satisfactory and/or as more satisfactory than a starting schedule), several complex and interacting aspects have to be considered. A schedule can generally be defined as a set of associations between tasks or jobs and workers, persons, groups of persons/workers or anyone/anything able to carry a task. There can be hard constraints associated with the situation, for example, a worker may not have more than *n* tasks associated to it, a worker can only be associated a task if they have the minimum required skills for carrying out the task, etc. Such hard constraints define what a feasible schedule is and what an unfeasible schedule is amongst the universe of possible (feasible and infeasible) schedules: a schedule that violates at least one hard constraint will be considered an infeasible schedule while all other schedules will be considered feasible schedules. As the skilled person knows, for a schedule to be a suitable schedule, the schedule must be a feasible schedule such that the search for a suitable schedule should search a suitable schedule amongst the feasible schedules and should not select an infeasible schedule. At the same time, the suitable schedule should attempt to optimise some aspects as much as possible. Examples of aspects that the schedule may attempt to optimise may include for example any of: a number of soft constraints violated (to be reduced), a number of consecutive tasks allocated during a worker's shift (to be increased so as to reduce the number empty slots), an amount of overtime (to be reduced), a travel distance (to be reduced), etc. These aspects to optimise can be taken into account in one or more cost functions which should be minimised (or maximised) for the best schedules. The problem of finding a suitable schedule for a set of tasks and workers can therefore involve trying to identify a suitable feasible solution among the vast number of possible solutions for allocating the tasks to the workers, while trying optimise one or costs functions.

Another point which affects the complexity of identifying a suitable schedule is that neighbouring or similar schedules (schedules which are very close to each other in the allocation of tasks) may result in very different outcomes regarding costs and/or constraint violations. For example, while one schedule may not violate any hard constraint and have a relatively low cost compared to other schedules, making one minor change to this one schedule may then result in a new schedule with one or more hard constraint being violated and/or a cost suddenly increasing to an unacceptable value. As a result of this chaotic behaviour, conventional approaches for finding optimised solutions for simple problems (e.g. using dichotomy or searching for neighbours of relatively good solutions already identified) are not expected to be successful or helpful as they are expected to be more likely to miss potentially good schedules, for example schedules which may be remote for the schedules considered by the search.

Problems of this kind are classified in the domain of computation complexity theory as NP-complete and NP-hard, meaning that as the problem grows in size, the time taken to deterministically find ideal solutions increases exponentially. Consequently, for any real-life situation with different workers, tasks, hard and soft constraints and aspects to optimise, the computing resources and time required to find the best overall schedule are far too vast to make a search of the best schedule possible or realistic. Thus, heuristic approaches are used to find useable, rather than ideal, feasible solutions within an acceptable amount of time. Such approaches stop when a good enough solution has been identified or when the search has been running for a certain amount of time and the best solution amongst the solutions searched can be identified as the best solution that could be found. These approaches are generally designed to work for a specific problem and can be difficult to adapt for a different problem. In particular, in view of the “no free lunch theorem”, while a first search method may be optimised to address a specific set of problems and be expected to yield good results in a reasonable time with this set of problems, this first method may then be expected to be sub-optimal for another set of problems and a different searching method would have to be devised to optimise the identification of suitable feasible solutions to the other set of problems in a reasonable time. In view of the complexity and the nature of the search, even with the vast computing capabilities now available in computers and computer systems, an exhaustive search is clearly out of reach and while a computer implementation for such a search is considered as being essential for attempting to identify a suitable feasible solution in a reasonable amount of time, it is considered desirable to identify any means by which the computer implementation of such a search can be accelerated and/or simplified.

In other words, due to the difficulties that are faced when trying to improve a search for a suitable schedule, any improvements in the efficiency of the identification of suitable feasible schedules are generally challenging to identify. Also, in view of the complexity of such systems and of the exponential increase for any additional factor added, any such improvements can translate in very significant improvements in the time needed to find a suitable feasible solution and/or in the computing resources required for identifying a suitable solution.

## SUMMARY

The invention is defined by the appended claims.

According to a first example of the present disclosure, there is provided a method for encoding first schedule data into a data structure for use in a quantum annealing optimisation process. The method comprises determining, by a process optimisation computing device, a schedule indicator value for each of a plurality of schedule data entries based on whether each of a first set of recorded tasks has been allocated to one or more of a plurality of identifier data names in one or more of a first set of time periods in accordance with the first schedule data to generate schedule portion data; determining, by the process optimisation computing device, a hard constraint indicator value for each of a plurality of hard constraint data entries based on whether at least one of a plurality of hard constraints has been violated by the allocation of the first set of recorded tasks to one or more of the plurality of identifier data names in one or more of the first set of time periods in accordance with the first schedule data to generate hard constraint portion data; and generating, by the process optimisation computing device, the data structure for use in a quantum annealing optimisation process based on the determined schedule portion data and the determined hard constraint portion data, thereby encoding the first schedule data into a data structure for use in a quantum annealing optimisation process.

Accordingly, there is provided a method for generating a data structure and data which is suitable for use in a quantum annealing optimisation process and for encoding a schedule represented by schedule data. The schedule can thus be encoded in a data form which can be manipulated by a quantum annealing optimiser, e.g. a computer implemented quantum annealing optimiser.

The data structure may be a binary data structure, wherein the schedule indicator data entries are binary schedule indicator data entries and wherein the hard constraint data entries are binary hard constraint data entries. As will be clear from the discussion below, in addition to provide a method for encoding a schedule into a data structure such that it can be optimised, by encoding the schedule into a binary data structure, the schedule can be encoded to facilitate a quicker optimisation implementation.

The hard constraint data entries may indicate, for each identifier data name of the plurality of identifier data names and for each hard constraint of the plurality of hard constraints, whether each of the plurality of hard constraints has been violated by the task allocations for each of the plurality of identifier data names in accordance with the first schedule data.

The method may comprise determining, by the process optimisation computing device, a task assignment indicator value for each of a plurality of task assignment data entries based on whether each of the first set of recorded tasks has been allocated to one of the plurality of identifier data names in accordance with the first schedule data to generate task assignment portion data, wherein the generating the data structure is further based on the task assignment portion data.

The method may comprise determining, by the process optimisation computing device, a conflict indicator value for each of a plurality of conflict data entries based on whether each the plurality of identifier data names has been allocated two or more of the first set of recorded tasks in one or more of the first set of time periods in accordance with the first schedule data to generate conflict portion data, wherein the generating the data structure is further based on the conflict portion data.