POLITECNICO DI TORINO

POLITECNICO DI TORINO

Master of Science in Computer Engineering

Master Thesis

Design of a parallel hardware architecture for Quantum Annealing Algorithm acceleration

Advisors
Prof. Andrea Acquaviva
PhD Gianvito Urgese

Evelina Forno

July 2018

Abstract

Optimization problems can be encountered in many fields of science and technology. To solve such problems means to select a best solution out of an exponentially large set of discrete candidates; as the size of the problem increases, so does the solution space, and brute-force techniques quickly become impracticable in the search for an exact solution. In fact, many combinatorial optimization problems are NP-hard in computational complexity. While exact solvers exist for several NP-hard problems, resource utilization and computation times continue to be an issue. For most practical applications, it is not necessary for the solution to be exact, as long as it is of good enough quality. Research has then turned to heuristic solvers, which guarantee a minimum quality of solution, while performing computation in a deterministic time. These characteristics make them desirable as general-purpose solvers as well as for safety-critical and real-time applications.

Many commonly-used solvers are metaheuristic algorithms that borrow ideas from the world of physics and biology. Among these is Simulated Annealing (SA), which utilizes the concept of slow cooling of a disordered system to explore the solution space until settling at a minimal energy state when the temperature parameter reaches zero. Quantum Annealing (QA) is an emerging technique derived from SA, originally proposed in the field of quantum computing. It attempts to exploit quantum entanglement to explore the solution space as a superposition of all possible system states at once. While an effective implementation of this type of quantum computer has yet to be proven, Simulated Quantum Annealing (SQA), an algorithm derived from QA theory and designed to run on classical computers, has found success as a metaheuristic. SQA simulates quantum superposition by adding one more dimension to the SA simulation space where a finite number of copies of the system, called *replicas*, exist. Entanglement among states is approximated by adding a correlation factor between replicas that has a function equivalent to that of temperature in SA; as the annealing goes on, the correlation between neighbor quantum states increases, eventually causing all replicas to collapse into a single optimal solution.

Studies have shown that SQA can be applied to solve NP-hard problems with faster convergence and better quality of result than other traditional heuristics.

However, due to the presence of several replicas, the computation cost of this algorithm is quite higher than that of Simulated Annealing, requiring higher amounts of memory and much longer execution times. At the same time, because of the population-based nature of the algorithm, SQA is particularly suitable to parallelization. In this context, I decided to study the feasibility of QA for the fully-connected Ising model; results led me to concentrate instead on the Multidimensional Knapsack Problem, which proved a better target for QA. I designed a parallel hardware accelerator for the solver algorithm, partially developed during my internship in NEC Japan. The various phases of the project are detailed in this thesis.

The Ising spin glass model is a mathematical model of ferromagnetism used in statistical mechanics. There are several variations of the Ising model depending on the connectivity between spins, which determines the geometry of the spin glass. My first examination of the performance of QA, using the 2-dimensional spin glass problem, confirmed literature results and showed definite improvement over SA. However, the most interesting version of the Ising spin glass model for practical application is the fully-connected model; in fact, mappings exist for many NP problems to the fully-connected spin glass model, and a solver for this model may be considered a universal solver for all such problems. A straightforward implementation of the fully-connected spin glass model is impracticable: the complexity of the interaction is exponential in the number of spins, and simulation of one step takes long computation times; most importantly, since spin updates are deeply correlated, parallelization is an arduous task. We considered then a minor embedding scheme in the form of a Chimera graph, the same structure realized in some hardware quantum annealers. I found that such a scheme was not effective when running SQA, as the additional ferromagnetic couplings required to ensure coherency of the Chimera graph interfere heavily with the Trotter coupling driving the annealing. As such, the realization of a general-purpose QA solver based on fully-connected Ising with spin-flip dynamics is not possible without significant modifications to the algorithm.

I turned then my attention to ad-hoc solvers. I selected the Multidimensional Knapsack Problem (MKP), an NP-hard resource allocation problem with applications to computer task scheduling, network data packing, financial investment planning and logistics. I implemented three software solvers in C++ for Simulated Annealing, Quantum Annealing, and a hybrid PSO-genetic optimization algorithm. I also implemented a variant of Quantum Annealing, Restricted Quantum Annealing (RQA), which strives to improve on the results of QA by restricting the search space: partial solutions that are found multiple times during annealing are locked and assumed to be part of the final solution. All algorithms were tested on the SAC-94 and Chu-Beasley benchmarks, evaluating for each the mean absolute percent error, the least error, the standard deviation of results and the CPU-time. The results suggest that the efficacy of QA is indeed dependent on the problem parameters, as the Weingartner set of problems in the SAC-94 returned particularly poor

solutions; however, QA delivered better quality of result in all the ORLib benchmarks. RQA outperformed other methods more significantly as the problem size grew, with the large, strongly-constrained problems in the Chu-Beasley benchmark showing the most benefit. The increase in computation time due to the complexity of QA is significant, although when compared to the PSO, it scales better with respect to the number of constraints.

I described the hardware implementation of the MKP-RQA solver behaviorally in SystemC and performed high-level synthesis with the *NEC CyberWorkBench HLS compiler*, then executed logic synthesis with Altera Quartus II for a Stratix V FPGA kit. The system consists of an array of 16 processors, each representing a Trotter replica for Quantum Annealing. The QA algorithm for solving MKP is implemented within each processor as a finite state machine; copies of the problem parameters are locally stored within each processor, and random number generation is achieved with a 32-bit LFSR. I fine-tuned the behavioral description of the algorithm with calculation simplifications and approximations to avoid multipliers and exponential functions, which were replaced with shifters and look-up tables. Each replica keeps its current solution in a register shared with its immediate neighbors. A controller module ensures synchronization of the processors at every simulation step. Since typical simulations for large problem instances can count millions of steps, the precalculated 16-bit values for the coupling parameter at each simulation step are offloaded to a fast on-chip memory connected to the design via a bus bridge with an Avalon-Memory Mapped interface; the controller activates a pair of buffers that alternately read from memory and make the data available to the processors. The controller also receives the current fitness value from each replica and detects when a new optimal solution has been found. Finally, an RQA engine module calculates the frequency of partial solutions and outputs a binary vector describing the locked items. Frequency calculation is performed with a SWAR algorithm for popcount to avoid computing long sums.

Since the problem size and parameters are hard-coded into the implementation, I performed synthesis multiple times for a variety of problem sizes. Synthesis results show that the total area of logic utilization grows linearly with the number of replicas and problem size in a deterministic way. As for the timing performance, the maximum frequency decreases for larger problem implementations; this is to be attributed to the exponential growth in number of interconnects and increasing length of the combinational paths. At the maximum frequency of 164 MHz, the FPGA solution is 150× faster than the software at executing the same number of Monte Carlo steps. Fitting was performed with the most aggressive settings for timing and the final area occupies 63% of the total logic. The power consumption estimates also show energy savings of 99.7% with respect to the execution of the same algorithm on CPU, while RTL simulations show that the hardware version can produce a similar quality of result as the software.

Contents

List of Figures 8

List of Tables 10

1 Introduction 11

2 Background 15
2.1 Optimization metaheuristics . . . 15
2.1.1 Simulated Annealing . . . 15
2.1.2 Genetic Algorithms . . . 17
2.1.3 Particle Swarm Optimization . . . 17
2.2 Quantum Annealing . . . 18
2.3 NP-hard Optimization Problems . . . 20
2.3.1 The Ising spin glass model . . . 20
2.3.2 The Multidimensional Knapsack Problem . . . 22

3 Materials and methods 25
3.1 Software implementations . . . 25
3.1.1 Quantum Annealing of the Ising spin glass model . . . 25
3.1.2 Quantum Annealing of the Multidimensional Knapsack Problem . . . 28
3.1.3 Other implementations . . . 30
3.2 FPGA architecture . . . 31
3.2.1 MKP Processor . . . 33
3.2.2 Controller . . . 34
3.2.3 RQA Engine . . . 36

4 Results and discussion 37
4.1 Software testing results . . . 37
4.1.1 Ising spin glass model . . . 37
4.1.2 Multidimensional Knapsack Problem . . . 40
4.2 FPGA synthesis results . . . 47
4.2.1 Logic synthesis results . . . 47

4.2.2 Result analysis 48

5 Conclusion 55

A MKP Benchmark Results 57
A.1 SAC-94 benchmark 57
A.2 Chu-Beasley benchmark 61

Bibliography 83

List of Figures

2.1 The Quantum Annealing algorithm 19

3.1 Time evolution of 5 QA replicas solving a 2-dimensional 32×32 Ising spin system 27
3.2 Iterative embedding schema for the Chimera graph [31] 29
3.3 Block diagram of the FPGA architecture 32
3.4 Block diagram of the interconnection between the solver module, the Avalon-MM bus, and the on-chip RAM 32
3.5 The MKP processor 33
3.6 Block diagram of the J_t multiplication stage 34
3.7 Block diagram of the Controller module 35
3.8 Block diagram of the RQA engine 36

4.1 Performance comparison of SA and QA on the 2d Ising model . . . 38
4.2 Performance comparison of SA and QA on the 3d Ising model . . . 39
4.3 Performance comparison of SA and QA on the fully connected Ising model with Chimera 40
4.4 Performance comparison of SA, QA and HPSOGO on the SAC-94 library 42
4.5 Standard Deviation for SA, QA and HPSOGO results on the SAC-94 library 43
4.6 Computation times for SA, QA and HPSOGO results on the SAC-94 library 43
4.7 Performance of SA, QA and HPSOGO on the ORLib library 44
4.8 Standard deviation for SA, QA and HPSOGO results on the ORLib library 45
4.9 Computation time for SA, QA and HPSOGO results on the ORLib library 46
4.10 Area estimation as a function of R 48
4.11 Area estimation as a function of C and N 50
4.12 Maximum frequency obtained by Quartus synthesis for varying hardware size. 50

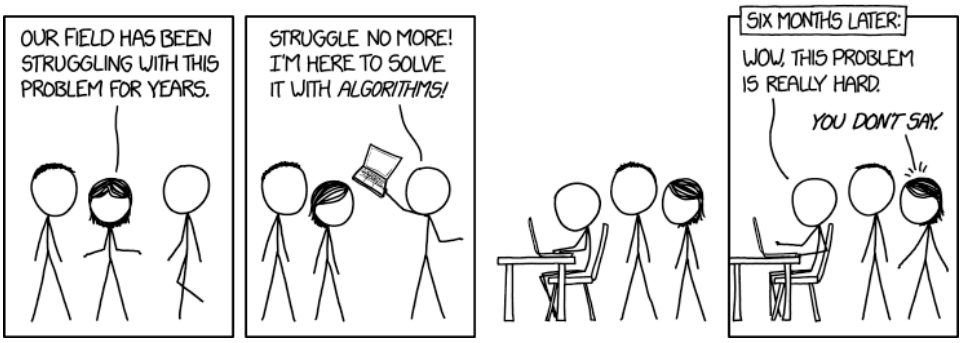
4.13 Comparison of results from the FPGA and software versions of the algorithm 51
4.14 Comparison of random trials per step as a function of τ, benchmark problem OR30x500-0.25. The regression function used for the FPGA estimation is $f(\lambda) \approx 13.17\lambda^{0.12}$. 52
4.15 Execution times for the FPGA and software versions for the benchmark OR30x500-0.25 53
4.16 Speed of convergence to result for the software and FPGA versions, benchmark problem OR30x500-0.25 53

List of Tables

4.1 Results of logic synthesis for the MKP-RQA module 47
4.2 Results of logic synthesis for the entire architecture 49

Chapter 1

Introduction



XKCD #1831 by Randall Munroe (CC BY-NC 2.5)

An optimization problem is the problem of selecting the "best" solution out of the total space of feasible solutions. This type of problem is often encountered across many fields of human activity; in computer engineering alone, we are familiar with the many optimization methods involved in the logic and physical synthesis of VLSI circuits, as well as issues such as task scheduling within operating systems and the optimal insertion of scan stitching registers for circuit testing. But this type of problem is also of paramount importance in fields such as finance and investments (capital budgeting [1], inventory management [2], portfolio optimization [3]), production and distribution (machine allocation [4], transportation model [5], choice of facility location [6]), human resources (employee scheduling [7], workforce composition [8], office assignment [9]).

The variables in an optimization problem can be continuous or discrete; solving a discrete optimization problem means to select a best configuration out of an exponentially large, but finite, set of solutions. Such problems are also referred to as combinatorial optimization problems. A combinatorial optimization problem [10] is formally defined as a 4-tuple (I, f, m, g), where I is a set of problem instances, $f(x)$ is the set of feasible solutions for instance x, $m(x, y)$ is the measure (or *fitness*)

associated with solution y, and g is the goal function, which can be a minimum or maximum. Every combinatorial optimization problem has an associated *decision problem*, a problem that has a "yes" or "no" answer, asking whether there exists a feasible solution for a given measure m_0. Combinatorial optimization problems for which the decision problem is in NP are called NP-Optimization problems; this means that there is no known algorithm that can solve them in polynomial time.

Indeed, these problems typically describe highly stressed systems. For example, *constrained* optimization problems express explicit constraints on the variables which can be modeled in variously complex ways, from a simple lower/upper bound to systems of equalities and inequalities. *Multi-objective* optimization requires finding a tradeoff between several conflicting parameters. As the number of constraints and variables increases, the solution space also inflates, and the sheer number of candidate solutions is impracticably large for brute-force methods and other basic techniques that are subject to combinatorial explosion; more sophisticated algorithms must be devised to explore the solution space in an intelligent and efficient way.

Exact software solvers have been investigated for several NP optimization problems, and their development and improvement is a popular subject in academic research. For example, the Traveling Salesman Problem (TSP) [11] [12] is one of the most intensely studied among such problems: it involves finding the least-distance circular tour of a set of cities. While the TSP has obvious applications in the realm of transportation and interconnection design, it has a worst-case time complexity of $O(2N^2)$, N being the number of cities in the graph. An exact solver, the Concorde TSP Solver, was developed in the late '90s based on the branch-and-cut method. This solver is widely regarded as the fastest exact TSP solver and, in 2006, it accomplished the feat of solving every benchmark in the TSPLIB benchmark library, the largest being `pla85900`, a tour describing the travel time of a laser drawing interconnections between 85,900 locations on a chip. Other exact solvers, such as CORAL for the Multidimensional Knapsack Problem [13] and MaxSolver for the Maximum Boolean Satisfiability Problem [14], have been proposed in literature, all utilizing efficient techniques such as dynamic programming, ILP and branch-and-bound.

However, while exact solvers are successful in finding the best solution, resource utilization and computation times continue to be an issue, as some of the largest known benchmarks can require years (or centuries) of CPU time to reach the exact solution. For example, the reported time to solution for `pla85900` by Concorde was 136 years; less dramatically, TSP instances of as few as 1,200 cities can take more than 1,000 seconds to solve. Additionally, problem instances that appear similar can require very different and unforeseeable amounts of computation time to solve: the `rl1304` benchmark takes Concorde only 189.20 seconds, while `d1291`, an instance of similar size, takes 27393.72 seconds. Another factor that can heavily influence computation complexity for exact solver is how constrained the problem is.

For the Multidimensional Knapsack Problem of 100 items, the CORAL developers reported computation times of $10 \sim 200$ seconds for instances with 10 constraints; when increasing the number of constraints to 30, time to solution runs up to 2,000 seconds.

All the above reasons explain the continued interest in heuristic solvers. For many practical applications, the exact best solution is not necessary, as long as an approximate solution of sufficient quality can be found. Good heuristic solvers guarantee a minimum quality of solution within few per mille of the exact solution; their computation time is generally deterministic, which is a desirable characteristic in safety-critical and real-time application where a valid solution is required to be produced in finite time. Since heuristic algorithms generally make use of stochastic or "blind" permutation, they are less impacted than exact solvers by the presence of a high number of constraints. Moreover, in situations where an exact solver is not already available, the development of a heuristic algorithm can be much faster and less expensive than that of an exact method.

Chapter 2

Background

There are a wide variety of approximate solvers for NP-Optimization problems. The ones presented in this chapter are classified as *metaheuristics*, that is, they are higher-level algorithms that implement mechanisms to choose between various strategies to explore the solution space. As such, the fundamental difference between them is the implementation of the "artificial intelligence" driving this choice; the inspiration behind this implementation has often come from observations in the fields of biology and physics, with algorithm designers trying to reproduce the innate efficiency of natural phenomena. Metaheuristic methods make extensive use of stochastic optimization for an effective exploration of the search space, and require very few assumptions *a priori* about the problem to be solved, therefore they are applicable to a wide range of problems with a few adaptations.

2.1 Optimization metaheuristics

In this section, we introduce three well-known metaheuristics. Simulated Annealing is one of the simplest and most used approximate solvers, and the inspiration for Quantum Annealing (QA); it has in common with QA the basic update mechanics. Genetic Algorithms and Particle Swarm Optimization are, like QA, population-based metaheuristics, but they employ fundamentally different update mechanics [15].

2.1.1 Simulated Annealing

Simulated Annealing (SA) [16] may be the most popular optimization metaheuristic. In fact, it is based upon the concept of annealing in metallurgy, itself one of the first optimization methods ever discovered by man: the process of heating, then slowly cooling down a metal alloy, inducing recrystallization and producing changes in the material's ductility and hardness.

The Simulated Annealing algorithm utilizes this concept of temperature as a measure of the freedom of movement in the solution space. A high value for the temperature parameter is set at the beginning of the annealing, allowing for strong global search.

At each iteration of the algorithm, a new random solution is generated. The fitness of the new solution, equivalent to the system Hamiltonian, is evaluated; if the new solution improves on the current one, the current solution is replaced, as in a local descent algorithm. Since local descent can easily cause the system to be stuck in local minima, a mechanism is needed to escape local energy wells; SA accomplishes this by applying at every step the Metropolis-Hastings algorithm. While lower-energy solutions are always accepted, there is a chance to stochastically accept a higher-energy solution and thus move to a potentially unexplored part of the search space, if

$$\rho \leq e^{\frac{-\Delta H}{T}} \tag{2.1}$$

where ΔH is the energy difference between solutions, T is the current temperature, and ρ is a randomly generated number in the range [0,1]. The probability of choosing a worse solution becomes lower and lower as the annealing continues, and at the end of the process the system ideally converges to a global minimum.

SA is well-tested, efficient and robust, but it can require long computation times to converge to an acceptable solution for large problem instances. It is widely used in the VLSI field for the generation of connection paths, placement and other optimization. For example, it has been successfully employed in solving floorplanning [17] and placement [18] problems, which are classified as NP, obtaining better results than other types of heuristics such as PSO and Ant Colony. An outstanding example is the use of SA in placement algorithms within commercial software such as Quartus II [19] for FPGA design optimization. Research within the field has led to several improvements to SA, especially in parallelization efforts, which yield better results and linear speedups with respect to the classic algorithm while trying to balance the added complexity in synchronizing several solver processes.

There has been interest from several research teams in developing parallel hardware architectures for accelerating SA on FPGAs and GPU. Since SA is an inherently sequential algorithm, workarounds are necessary to exploit concurrency; solutions have been proposed both on CPU [18] [19] and GPU [20] which run independent SA solvers in separate threads, then choose the best solution reached among all threads. A similar approach has been attempted on FPGA [21], reporting success for relatively small problems (1024-bit, corresponding to a 32-city TSP problem). FPGA acceleration of Monte Carlo solvers, which apply concepts compatible with SA, has found great success in physical simulations of the nearest-neighbor Ising spin glass such as the *Janus II* computer [22]; however, most NP problems require higher levels of connectivity.

2.1.2 Genetic Algorithms

Genetic Algorithms (GA) [23] are a family of metaheuristics inspired by the inheritance and random mutation of genetic characteristics. At first, a population of chromosomes is generated, each representing a candidate solution. The genetic simulation utilizes the se*lection*, *crossover* and *mutation* operators. The evolution of the group of chromosomes is simulated as an iterative process; each step represents a generation, where the fitness of each solution is evaluated (using the cost function of the problem) to select the parents of the next generation. This *selection* operator can be implemented in different ways [15]; selection methods generally involve stochastic sampling weighted by solution fitness.

In the *crossover* step, parts of the parent chromosomes are randomly swapped to create new solutions. Various *crossover* implementations [15] define different size, number and choice of boundary for the swapped sections.

Finally, the *mutation* operator introduces random changes in the chromosomes in an effort to avoid getting stuck in local optima.

As generations succeed one another, the overall population tends to evolve towards solutions of higher fitness.

GA is flexible and easy to extend by combining it with other algorithms. However, it exhibits slow convergence rates, and the quality of result is difficult to control due to the stochastic nature of crossover and mutation [15].

2.1.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [24] is a heuristic based on swarm intelligence. The system consists of a population of *particles*; each particle is assigned a position X in the solution space, corresponding to a candidate solution, and a velocity V that determines its current and future displacement across the search space. Each particle possesses information about its *Local Best* solution and the overall *Global Best* solution. At each iteration i, the next position X_{i+1} is computed for each particle:

$$X_{i+1} = X_i + V_i \tag{2.2}$$

The speed term is also updated at every step:

$$V_i = c_c \cdot r_1 \cdot dBLS + c_s \cdot r_2 \cdot dBGS \tag{2.3}$$

where r_1 and r_2 are two random numbers in the range [0,1]; $dBLS$ and $dBGS$ are the distance of the current solution from the local and global best, respectively; and c_c and c_s are two weights determining the attraction of the particle to the best known solutions.

PSO is popular for its easy and fast implementation and efficient global search; however its local search is weak with a slow convergence rate and it easily gets caught in local minima when solving complex problems [15].

2.2 Quantum Annealing

Quantum Annealing is an emerging metaheuristic technique, originally proposed in the field of quantum computing [25]. The algorithm is inspired by classical Simulated Annealing, but instead of applying a thermal gradient to the system, it applies a slowly diminishing transverse field. Real quantum computing-based solutions rely on the quantum-mechanical superposition of all possible candidate states with equal weights; the system is then allowed to evolve by the SchrÃűdinger equation and explore the state space. As the strength of the transverse field reduces to zero, the system collapses into the ground state of the classical Ising model describing the Hamiltonian of the optimization problem. The algorithm performed by hardware quantum annealers such as those produced by D-Wave Systems is generally referred to as *Adiabatic Quantum Computation*, because the system stays close to the Hamiltonian ground state during evolution with a slow rate of change in the transverse field.

The quantum annealing process can be simulated in a traditional computer using stochastic techniques variously referred to as *Quantum Monte Carlo, Simulated Quantum Annealing, Quantum Stochastic Optimization,* and so on. This class of algorithms involves an adaptation of the classical Metropolis-Hastings algorithm (also used in Simulated Annealing) to step out of local optimum solutions [26]. Despite generating much enthusiasm in the scientific community for being one of the first practical realizations of quantum computers, hardware quantum annealers have generated just as much criticism [27]; the results of experimental runs so far have failed to outperform classical computers in optimization problems. On the contrary, Simulated Quantum Annealing, while drawing criticism for failing to accurately model the mechanics of Adiabatic Quantum Computation, is proven to be competitive as a classical metaheuristic: studies have shown that SQA can be applied to solve NP-hard problems with faster convergence and better quality of result than other traditional heuristics.

What sets apart Quantum Annealing from Simulated Annealing is the emulation of the *quantum tunneling* effect for escaping local minima. The key parameter Γ, which indicates the strength of the transverse field, represents the quantum tunneling width and determines the radius of local search. At first, the neighborhood comprises the whole search space; during the annealing this radius is gradually reduced.

SQA simulates quantum superposition by adding one more dimension to the SA simulation space where a finite number R of copies of the system, called *replicas*, exist. Entanglement among states is approximated by adding a correlation factor J_t between replicas that has a function equivalent to that of temperature in SA; as the annealing goes on, the correlation between neighbor quantum states increases, eventually causing all replicas to collapse into a single optimal solution.

Figure 2.2 represents the flow of the Quantum Annealing algorithm. On the left,

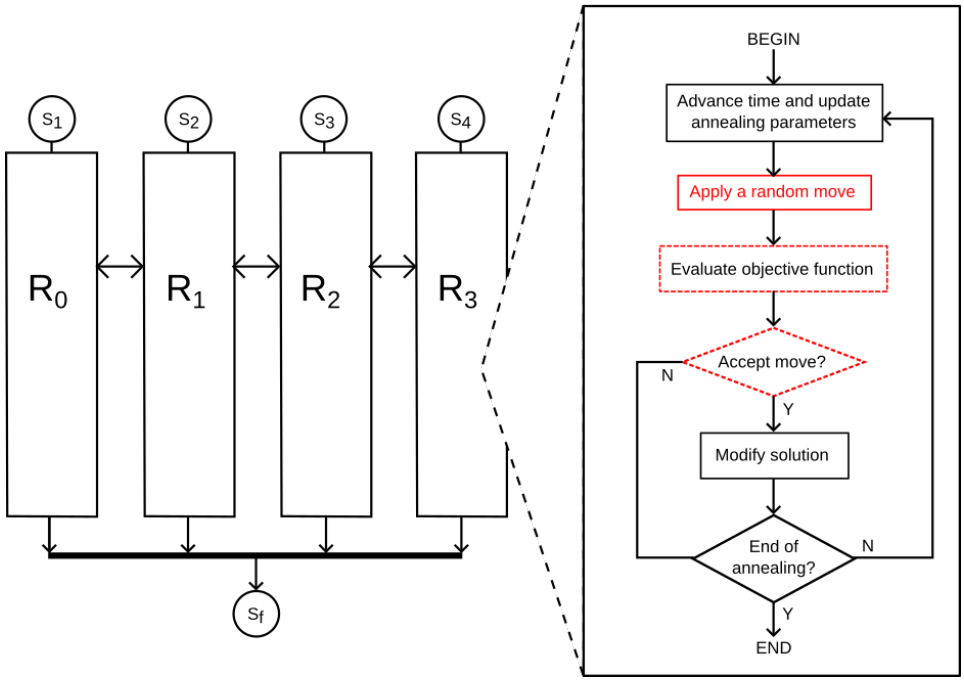


Figure 2.1: The Quantum Annealing algorithm

we have a visual representation of replicas as a series of parallel threads exchanging information with neighbors about their local solution; at the end of annealing, all replicas shall converge to the same solution. On the right, the flowchart of the algorithm executed within each replica: highlighted in red are the portions of the program that can be rewritten to fit different problem types (e.g., to allow only legal moves within the constraint system). The red dashed line indicates parts that stay the same, save for the problem Hamiltonian.

Due to the presence of several replicas, the computation cost of this algorithm is quite higher than that of Simulated Annealing, requiring higher amounts of memory and much longer execution times. At the same time, because of the population-based nature of the algorithm, SQA is particularly suitable to parallelization.

It should be noted that QA is not just a Simulated Annealing with R copies running in parallel. Normally, SA is only able to pass to a neighboring state on the energy landscape in one step, by thermal transitions. However QA theory [26] says that, by adding the J_t coupling, the model will be able to tunnel through energy barriers, avoiding local maxima, and exploring the state space more effectively. This can explain the faster convergence of Quantum Annealing.

In particular, thermal transition probability between energy states for Simulated Annealing is proportional to $e^{\frac{-\Delta}{k_B T}}$ (where Δ is the height of the energy barrier, k_B is the Boltzmann constant, and T the annealing temperature). This probability is dominated by the height Δ of the barrier, which means it is difficult to get out of a

very deep well of local minimum by means of thermal fluctuations. However, it has been demonstrated [28] that the quantum tunneling probability through the same barrier is proportional to $e^{\frac{-\sqrt{\Delta}w}{\Gamma}}$.

The tunneling probability depends not only on the height Δ, but also on the width w of the energy barrier. This means that QA shows significant advantage on problems where the energy landscape presents a high amount of perturbation with many high and thin barriers ($w \ll \Delta$). Indeed, the search of the ground state for an Ising spin glass model is one of these problems: since many NP problems can be formulated through the Ising model [29], we can apply QA to them and expect favorable results.

Quantum Annealing, however, should not be considered a panacea for all difficult problems. For example, based on the above analysis, it is evident that QA displays poor performance when the energy landscape presents wide and low energy barriers. As such, success of QA in solving a problem depends heavily on the shape of the energy landscape.

2.3 NP-hard Optimization Problems

In this section are presented two target problems for QA. A natural target for optimization when considering Quantum Annealing is the Ising spin glass model, as this is the problem architecture implemented in real quantum annealers. The Ising model is a powerful tool for NP-Optimization, since mappings from various NP problems to Ising have been defined; a solver for the fully-connected Ising model would be a sort of universal solver. As an alternative approach, I also took into exam a particular NP-Optimization problem, the Multidimensional Knapsack Problem.

2.3.1 The Ising spin glass model

One of the first proposed applications of QA [25] was the Ising model, a mathematical model of ferromagnetism used in statistical mechanics. It consists of a system of up/down spins organized in a graph. Each spin has a given radius of neighbor spins that it is allowed to interact with. The model is described by the Hamiltonian

$$H_c = -\sum_{<i,j>} J_{ij} s_i s_j \tag{2.4}$$

where the N spins s_i can take the values ± 1. The interaction between spins s_i and s_j on lattice sites i and j is described by the exchange coupling J_{ij}. $< i, j >$ means that i and j are neighbor spins; the radius of neighborhood depends on the chosen model. When spins are not neighbors their interaction is $J_{ij} = 0$, therefore such pairs do not contribute to the Hamiltonian.

The effectiveness of Simulated Annealing in finding the ground state of the Ising model has been demonstrated in research [16]. Since then, the Ising model has been used [26] as a benchmark to measure the performance of SA's derivative algorithms such as Quantum Annealing.

Several variations of the Ising spin glass model exist, based on the radius of neighborhood considered for each spin when computing the Hamiltonian. We have investigated a few of them, in the interest of evaluating their usefulness in solving optimization problems and their feasibility of implementation in FPGA.

The 2-dimensional spin glass model

The 2d spin glass has the spins arranged in a planar lattice. It is a nearest-neighbor (also known as Edwards-Anderson) model, usually considering only 4 neighbors for each spin (north, south, west, east). Periodic boundary conditions are applied.

The 2d model can be heavily parallelized on FPGA [30]. In fact, due to its high granularity, the update of each spin can be evaluated independently in each MCS. The lattice is usually partitioned with a checkerboard scheme to ensure that no neighboring spins are updated at the same time. For the nearest-neighbor model with 4 neighbors per spin, only 2 partitions are necessary. This means that a model of N spins can be realized with $\frac{N}{2}$ independent processors, attaining a full update (MCS) in 2 clock cycles. Many FPGA realizations of this architecture exist; the most important is the *Janus II* supercomputer [22] for the Monte Carlo simulation of 2d Ising systems, consisting of a modular array of FPGAs.

The 3-dimensional spin glass model

The 3d spin glass model is identical to the 2d model, but the spins are arranged in a cubic lattice. It is also a nearest-neighbor model, generally considering 6 neighbors (north, south, west, east, front, back).

The 3d model can also be heavily parallelized on FPGA with little extra cost compared to 2d. If we consider the same number of spins, the only overhead is more complex connectivity due to the increased size of the neighborhood.

The 3d model is more appealing than the 2d model because there is no algorithm that solves it exactly: therefore, a simulated heuristic that can provide an approximate solution is considered interesting. However, literature [22] [29] suggests that the usefulness of 3d spin glass models is limited to physics simulations, and there was no evidence that any NP optimization problems could be mapped to a 3d lattice effectively.

The fully-connected spin glass model

The fully-connected spin glass model is also known as the Sherrington-Kirkpatrick model. It is an Ising model with long (potentially infinite) range couplings, where

any two spins can be aligned with a ferromagnetic or antiferromagnetic interaction.

This model is very interesting because many NP problems (e.g. partitioning, covering and packing, knapsack, coloring and Hamiltonian cycle problems) can be mapped [29] to a fully or partially connected graph. However, simulation of the fully connected graph takes exponentially long computation times and is difficult to parallelize: since every spin flip decision depends on the status of every other spin, bonds must be evaluated sequentially to ensure the correctness of the system's evolution in time.

An FPGA architecture able to perform simulated annealing of the fully connected spin glass was introduced [21] in 2017. As a workaround to the interdependency of spin updates, it proposed to parallelize spin update calculation at every step, then execute only the spin flip that produces the best ΔH. It reported success for relatively small (1024-spin) problems, but such a solution may be difficult to scale to larger problems or to accommodate multiple replicas for QA. Additionally, the fact that only one spin is updated at every step might make computations long for large problem instances.

Minor embedding

Minor embedding is an alternative technique for performing annealing of fully-connected models. It consists of mapping the fully-connected model onto a graph with lower connectivity. This mapping is obtained by mapping every spin to a chain of spins on the lower-order graph in such a way that chains have complete connectivity; spins in the same chain are kept coherent by an interaction factor J_k.

The Chimera graph, a minor embedding scheme with fully-connected subgraphs of order 8, is used on the D-Wave quantum annealer [31]. Therefore embedding procedures for many problems' data structures have already been developed.

I have considered the Chimera graph for an FPGA application because the realization would be similar to 2d and 3d models, with slightly larger connectivity and delay overhead. Unlike the actual fully-connected model, it could be implemented by single spin flip dynamics. Moreover, this model is quite powerful since many NP optimization problems can be mapped to it. However, it should be noted that embedded problems have intrinsic embedding overhead, i.e., the time needed to encode the correct embedding of the problem, which is typically not trivial.

2.3.2 The Multidimensional Knapsack Problem

The Knapsack (or Rucksack) Problem [32] is a well-known problem in combinatorial optimization. Given a set of items, each with an associated *weight* and *value*, the solver is requested to select the items to insert in the knapsack such that (a) a maximum weight is not exceeded and (b) the total value of the collection is maximized. It is a classic resource allocation problem that identifies a number of variants

by imposing or removing a limit on duplicate items, multiple constraints, etc. The Multidimensional Zero-One Knapsack Problem (MKP) is one such variation, where only one copy of each item is allowed and multiple weight constraints must be met.

Definition

Given:

- n : number of items to pack,
- U_i : vector of item values (size n),
- M_j : vector of constraints on item weights (size c),
- $[p_{ij}]$: matrix of positive weights (size $n \times c$), where p_{ij} represents the cost of inserting item i with respect to constaint j,

Find an assignment $x = (x_i)_{1 \leq i \leq n} \in \{0,1\}^n$ such that the total value $\sum_{i=1}^{n} U_i x_i$ is maximised, while respecting the constraint set: $\sum_{i=1}^{n} p_{ij} x_i \leq M_j, \forall j \in [\![1, c]\!]$.

Real-life applications of the MKP include:

- Task scheduling problems, such as residential task scheduling under dynamic pricing for smart power grid technologies [33];
- Packing problems such as the placement of virtual machines in a cluster of physical machines for cloud computing [34];
- The optimization of financial operations such as securitization [35];
- Resource allocation for grid computing [36];
- Project portfolio management to optimize work practices within a business or enterprise [37].

A realization of a solver for the one-constraint 0-1 Knapsack Problem has been proposed on FPGA [38], but a hardware implementation for the MKP does not seem to have been attempted yet. Meanwhile, an exact solver of the single-constraint KP on GPU [39] reported a speedup of 26× over CPU for large instances.

Chapter 3

Materials and methods

3.1 Software implementations

3.1.1 Quantum Annealing of the Ising spin glass model

Heuristic simulation of the Ising model can be performed by the Monte Carlo method, which relies on a combination of deterministic computation and random sampling. The Monte Carlo simulation of the Ising spin glass consists of iterating over every spin and performing an update, i.e., deciding whether or not to flip each spin based on the status of its neighbors and the strength of its interaction with them. A Monte Carlo step (MCS) is concluded when all the spins in the systems have been updated.

The basis of QA implementations for the Ising model is the approach described in [26]. The cost function for the problem is derived from the quantum Hamiltonian via path-integral representation. To perform QA of Ising spin glasses, an additional term is added to the Hamiltonian by applying a transverse magnetic field Γ:

$$H_q = -\sum_{i<j}^{N} J_{ij}\sigma_i\sigma_j - \Gamma(t)\sum_{i}^{N}\sigma_i \tag{3.1}$$

Γ starts out at a high value and is gradually reduced to zero during the annealing.

In computer-simulated QA, the quantum effect is simulated by mapping the partition function of the quantum Ising model to that of a classical Ising model in one higher dimension, called imaginary time dimension or Trotter dimension (as the Suzuki-Trotter expansion is used to perform this mapping). This means that the system is simulated simultaneously in R different iterations or replicas, which start out completely independent but have a correlation factor J_t that grows in time, forcing them to converge to a single solution at the end of annealing.

J_t is calculated each MCS as a function of Γ:

$$J_t = -\frac{1}{2}\log(\tanh(\Gamma)) \tag{3.2}$$

Note that equation 3.2 is the function I have settled on for my implementation, but there are several variations in the definition of J_t across QA literature.

The 2-d and 3-d models

Both the 2-d and 3-d model include periodic boundary conditions along the imaginary time direction. The parameters of the problem are expressed by the matrix $[J_r]$ of random couplings between neighboring spin sites. The spin lattice is represented by a 3-d (for the 2-d model) or 4-d (for the 3-d model) array of integer values, randomly initialized to +1 or -1, identifying the direction of the spin. During one Monte Carlo step, the algorithm iterates over every spin in the system and attempts to perform a spin flip: the difference in energy for the potential spin flip is computed, and if the flip is advantageous, it is performed. Otherwise, the Metropolis-Hastings algorithm is performed to allow random acceptance of worse solutions.

An additional improvement to the algorithm is brought by adding global updates that flip a whole l-directed chain of spins occupying the same position, if it improves the solution.

Listing 3.1: Pseudocode for QA of the Ising spin glass

```
for (s = 0; s < τ; s++) {
    Update Γ with linear schedule: Γ = Γ_0 · (1 − (s − 1/τ));
    Update J_t: J_t = −1/2 log(tanh(Γ));
    for (every spin i in the lattice) {
        for (every replica l) {
            ΔH_pot = 2 · (Σ_{j∈neighbors(i)} J_r^{i,j} · s_j^l) · s_i^l;
            ΔH_kin = 2 · J_t · (s_i^{l−1} + s_i^{l+1}) · s_i^l;
            ΔH = ΔH_pot + ΔH_kin;
            rand = generateRandomNumber();
            if (ΔH_pot < 0 || ΔH < 0 || rand < e^{−ΔH}) {
                flip spin: s_i^l = −s_i^l;
            }
        }
        δH_chain = Σ_{l=1}^{R} ΔH_pot^{i,l};
        if (ΔH_chain < 0) {
            flip all spins in the chain;
        }
    }
}
```

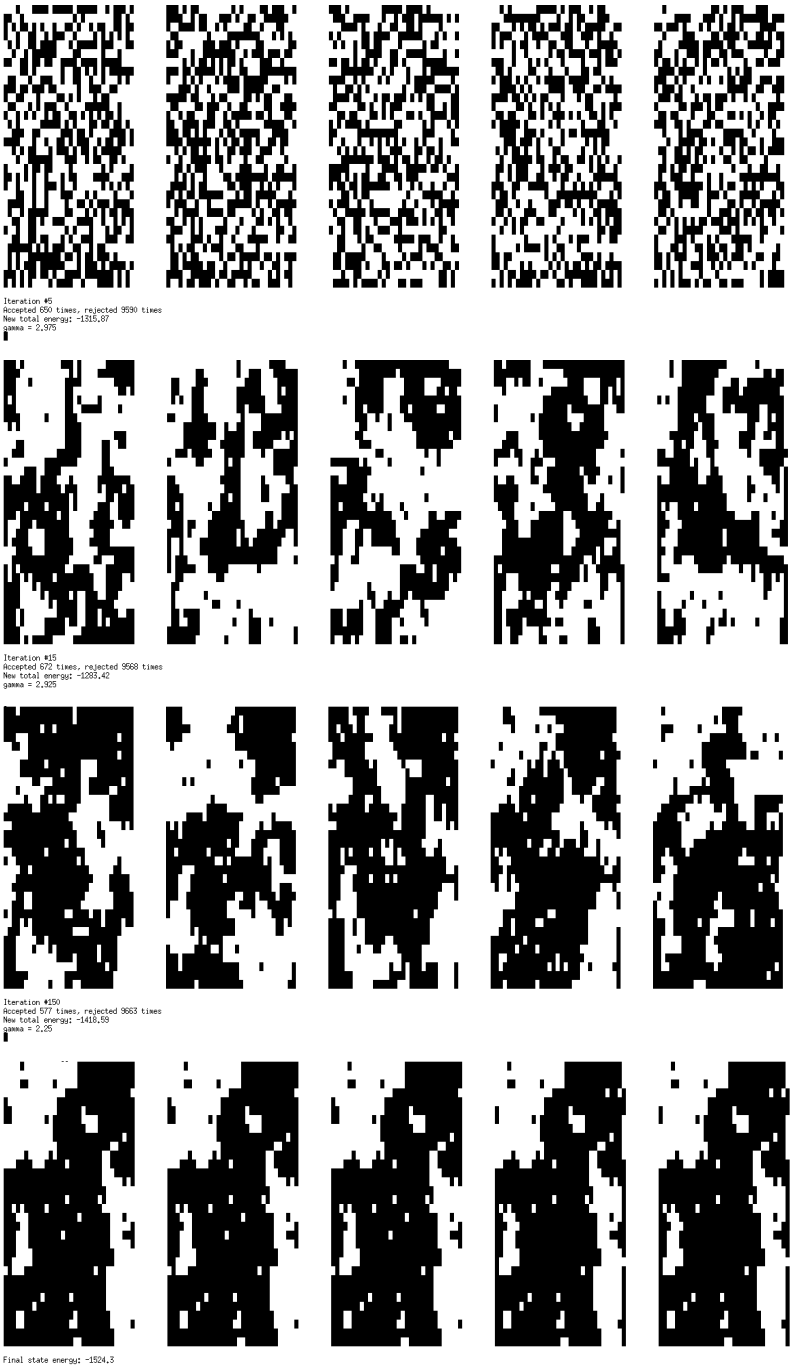


Figure 3.1: Time evolution of 5 QA replicas solving a 2-dimensional 32×32 Ising spin system

The fully-connected model

In the fully-connected model the spins are not organized in a lattice; random couplings are realized between every pair of spins, so there is no concept of neighborhood of a spin. However, QA replicas are still organized in a nearest-neighbor topology with periodic boundary conditions. The algorithm remains the same as detailed in figure 3.1, the only difference being in the calculation of ΔH_{pot}:

$$\Delta H_{pot,SK} = 2 \cdot (\sum_{j \neq i}^{N} J_r^{i,j} \cdot s_j^l) \cdot s_i^l \tag{3.3}$$

Since the sum now runs over $N - 1$ spins, computations are considerably slower than in the lattice graph.

The Chimera graph

Minor embedding with the Chimera graph can be implemented with the iterative embedding [31] procedure. Each of the spins (logical bits, or LB) in the original graph is mapped to 8 qubits in the Chimera graph following the modular triangle scheme shown in figure 3.2. The 8 qubits representing the same LB are tied by a nearest-neighbor ferromagnetic coupling J_F with negative weight, which serves as a sort of penalty function keeping the qubits in the chain coherent. The remaining connections are used to model the original connections between LBs.

The energy difference equation becomes

$$\Delta H_{pot,i} = 2 \cdot (\sum_{d \in Chimera-nb} J_r \cdot s_d + \sum_{p \in LB-nb} J_F \cdot s_p) \cdot s_i \tag{3.4}$$

where $Chimera - nb$ identifies the fully-connected neighbors in the same Chimera graph, and $LB - nb$ the neighbors in the LB chain, of spin s_i. The rest of the algorithm remains identical to the one described in listing 3.1.

3.1.2 Quantum Annealing of the Multidimensional Knapsack Problem

The representation for the Multidimensional Knapsack Problem is not dissimilar to the Ising spin model. Every replica works on a different knapsack and the knapsack is represented by an N-spin integer vector, where N is the number of items; the spin representing each item is +1 if the item is in the bag, -1 if not. It has been demonstrated [32] that it is possible to use the simplified path-integral Hamiltonian by only allowing moves that take our system to another valid configuration, i.e., only inserting and swapping items that do not violate any of the multidimensional constraints. This is achieved by starting with an empty knapsack (all spins

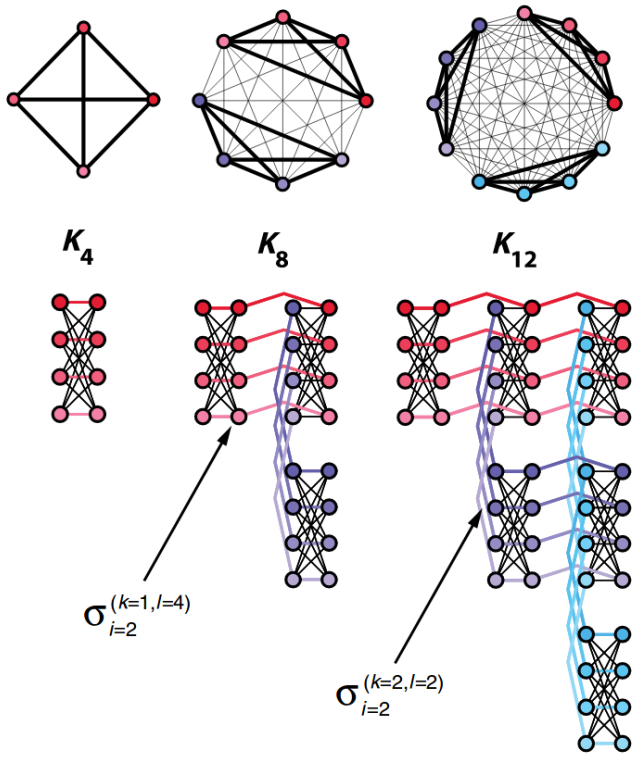


Figure 3.2: Iterative embedding schema for the Chimera graph [31]

= -1), adding random items until constraints are exceeded; then, all subsequent trials require swapping an item in the bag for a random one outside, provided that constraints are still met. The classical portion of the Hamiltonian is modified as:

$$H_{pot,MKP} = \sum_{i=1}^{n} U_i x_i \tag{3.5}$$

that is, the total value of the items added to the knapsack.

I applied two modifications to the existing algorithm:

1. The proposed mutation is to try to exchange an item a outside the bag with an item b inside the bag; if the exchange is not possible, "step back" by removing item b from the bag. Instead of doing this unconditionally, I run a new Metropolis-Hastings random trial to determine if b should be removed. This modification improves the convergence time of the algorithm and the quality of the result.

2. The value of J_t plays a paramount role in the annealing. However, no prescription for it is given in [32]. I made a few trials to find a proper value for J_t.

The main problem is that the available benchmark problems present a very broad range $(10 - 1000)$ of possible values for the item prices, and therefore a broad range of possible values for the potential energy ΔH_{pot}. Since stepping out of the local optimum depends on the value of $\Delta H = \Delta H_{pot} + \Delta H_{kin}$, it is evident that ΔH_{kin} must be roughly of the same order of magnitude as ΔH_{pot} for the Metropolis dynamics to work. ΔH_{kin} is directly proportional to J_t, therefore we want J_t to be of the same order of magnitude as the range of item prices.

One possible way to change the value of J_t is to change the value of Γ_0. However, modifying Γ_0 also influences the rate of growth of J_t. I identified the ideal rate of growth as that corresponding to $\Gamma_0 = 3.0$, as lower values cause the J_t interaction to spike to high values too early in the annealing (preventing the system from exploring the state space for most of the annealing), while high values make the growth of J_t too slow (greatly reducing the quantum tunneling effect).

A possible optimization for the MKP-QA solver is the Restrictive Quantum Annealing (RQA) paradigm proposed in [32]. RQA consists in disallowing the removal of elements that appear in a certain percentage of replicas (called the blocking frequency); it works on the assumption that if an edge appears in many partial solutions it is highly likely that it will be part of the final solution. In practice, the optimal blocking frequency varies depending on the benchmark problem, and it would be up to the user to run the algorithm with varying frequency values and find out the best value. However, as a rule of thumb, a frequency of 90% has been observed to be effective on a wide range of problems.

3.1.3 Other implementations

Simulated Annealing

Adaptations of the Ising model and MKP into Simulated Annealing are straightforward; they employ as cost function the classical part (ΔH_{pot}) of the Hamiltonian described in the QA implementations. The random moves are also performed in the exact same way. The threshold function for the Metropolis-Hastings algorithm is slightly different ($e^{\frac{\Delta H}{T}}$); temperature T is decreased with the same linear scheduled employed for Γ in QA.

Hybrid PSO-Genetic Optimization of the MKP

A novel and effective method for solving the MKP has been proposed in Hybrid PSO-Genetic Optimization (HPSOGO) [40]. This algorithm is a modification of the PSO with an extra step at every iteration where an ulterior optimization is performed on every particle using genetic operators. Once per iteration, two genetic

mutations are performed: a random number of positions of the item vector are copied first from the local, then from the global best solution. HPSOGO has reported great results on the SAC-94 benchmark, but it had not yet been tested on the entire Chu-Beasley set.

An important difference between HPSOGO and QA is that, while QA can only produce valid solutions, HPSOGO has an update strategy that also temporarily accepts solutions that violate constraints in order to explore the state space more aggressively. In practice this means that HPSOGO can often fail to find a valid solution at all. In fact, while results reported in literature showed that HPSOGO was able to find a ground state for many instances, its success rate was often lower than 100%.

In the interest of making a fair comparison between QA and HPSOGO, I chose a severe penalty function, giving solutions that violate constraints a negative total value. This ensures a 100% success rate while sacrificing some freedom of movement in the solution space.

3.2 FPGA architecture

This section illustrates the application chosen for development on FPGA: a parallelized Multidimensional Knapsack Problem solver with Restricted Quantum Annealing. The circuit takes as input the set of data representing the item profits, knapsack capacities and constraints, and it produces the solution as a vector of bits representing the items in the knapsack, together with its profit. I described this circuit behaviorally in SystemC and performed high-level synthesis with the NEC *CyberWorkBench HLS compiler* (CWB) [41], exporting the components to an RTL format.

The following nomenclature describes the MKP instances:

- Profit: the price of an item, equivalent to the potential energy contribution from the item. The goal of QA is to maximize the total profit of a knapsack.
- Weight: the cost of an item. In MKP an item can have multiple weights describing as many constraints.
- Capacities: the maximum sum of weights for each constraint allowed in the knapsack.
- Item vector: a binary vector of N bits describing the contents of the knapsack. The i^{th} bit is set if item i is in the knapsack.

The architecture (shown in Fig. 3.3) is composed of an array of R processors, each representing a Trotter replica for Quantum Annealing. Each replica shares its current item vector with its neighbors.

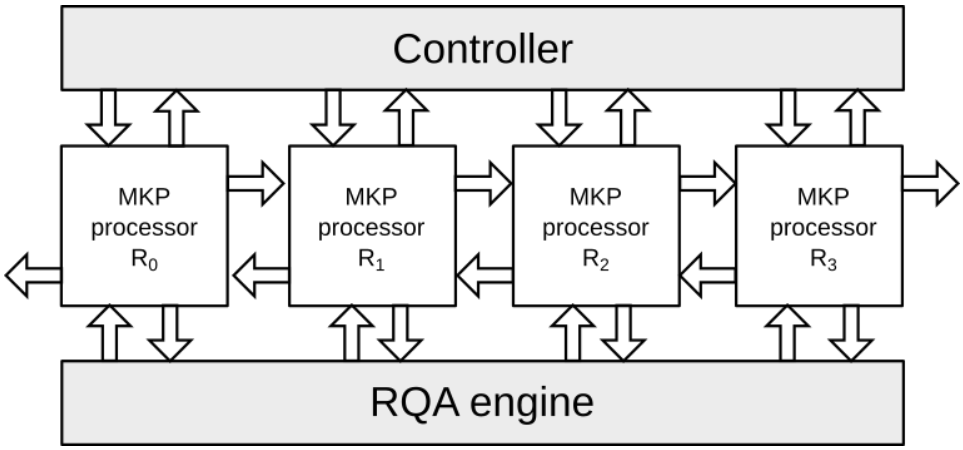


Figure 3.3: Block diagram of the FPGA architecture

There is also a *RQA engine* that receives the item vectors from the processors and calculates the frequency of each item across all solutions. It outputs a binary vector describing the locked items that replicas are no longer allowed to change.

Finally, there is a *Controller* module that ensures synchronization of the replicas during one Monte Carlo step (MCS). It fetches J_t values for the next Monte Carlo steps from an on-chip RAM, then enables the replicas to allow calculation of the next step and sends them the appropriate J_t value. It also receives the current *total profit* from each replica and detects when a new optimum has been found. The controller implements a Memory Mapped interface to the Avalon bus [42] for connection to the external RAM, as shown in Fig. 3.4.

In the following we will examine each module in detail.

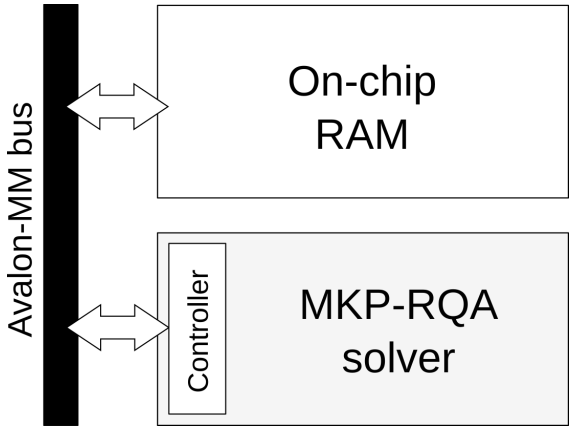


Figure 3.4: Block diagram of the interconnection between the solver module, the Avalon-MM bus, and the on-chip RAM

3.2.1 MKP Processor

The basic structure of the MKP processor is the same for all replicas. The core of this processor is a Finite State Machine that executes the operations for one Monte Carlo step of Quantum Annealing. Every MKP processor stores a copy of the problem data in local registers.

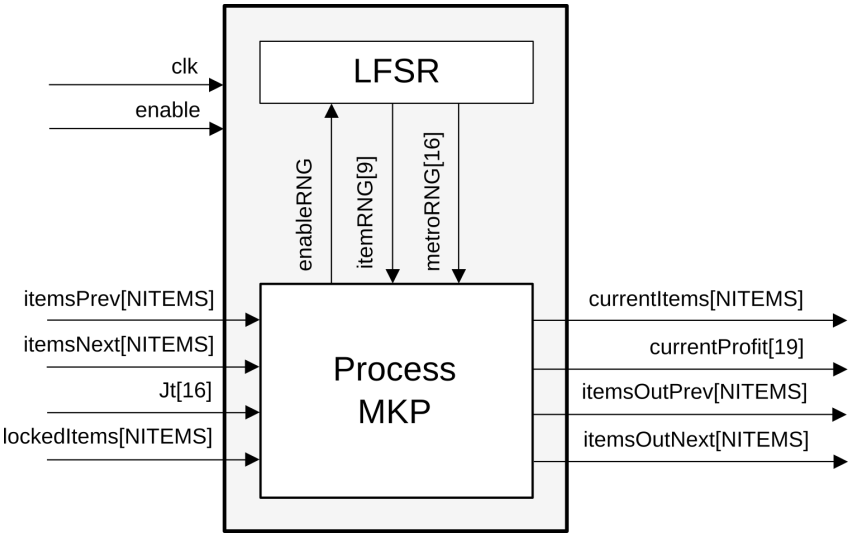


Figure 3.5: The MKP processor

The processor is a Finite State Machine of 23 states for the largest problem (30x500); because of a few branches in the algorithm, the average latency due to pipelining is lower than 23 cycles. However, since each random number generation attempt using the LFSR introduces an extra cycle of latency, the latency of this module in any given Monte Carlo step is not deterministic.

In fact, a key problem of implementing a stochastic algorithm on FPGA is the quality of the random number generation. I used a simple 32-bit LFSR, which provides good pseudorandom performance. From this LFSR up to 9 bits are selected as an item identifier (`itemRNG`) and 16 bits for the Metropolis random number (`metroRNG`). When the MKP solver needs a random number, it enables the LFSR and waits for the next clock cycle. Since the LFSR is a synchronous circuit, it necessarily introduces latency.

Most of the necessary instructions in the process datapath are adders, subtractors, and comparators. However, when we enter a Metropolis attempt to swap item A with item B, the calculation of the quantum portion of the Hamiltonian:

$$\Delta H_{kin} = J_t \cdot ((s_A^{l-1} + s_A^{l+1}) \cdot 2s_A^l + (s_B^{l-1} + s_B^{l+1}) \cdot 2s_B^l) \tag{3.6}$$

would introduce at least one 16-bit multiplier. The result of the right hand side parenthesis only has a few discrete possible values: $-8, -4, 0, +4, +8$. Then, the

ΔH_{kin} calculation can be accomplished by using exclusively LUTs and shifters, as shown in Fig. 3.6. This saves a considerable amount of area and reduces the critical path.

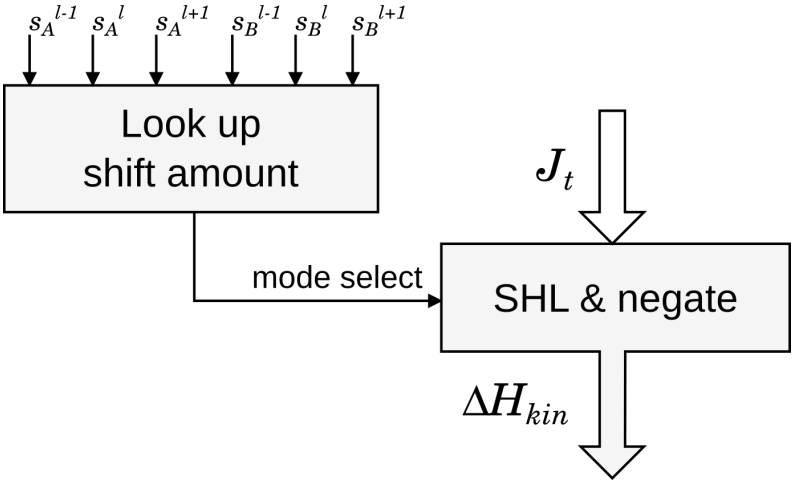


Figure 3.6: Block diagram of the J_t multiplication stage

The calculation of the exponential $e^{-\Delta H}$ for the Metropolis trial is also prohibitively expensive in FPGA, so I implemented it with a LUT indexed by ΔH_{kin}. I experimentally determined that a high precision is not necessary for this operation and the exponential LUT only needs 24 entries of 16 bits.

When it is necessary to evaluate ΔH_{kin}, each replica needs to have a stable copy of the item vectors from its neighbors, and these vectors should be from the same annealing step that the replica is currently in. FPGA implementations of the Ising model solve this problem by partitioning the spins into two groups and sharing each spin unit between two neighboring spins that are processed in separate clock cycles.

From simulation statistics I determined that calculation of ΔH_{kin} is not performed in 98-99% of annealing steps within a simulation. Then, the replicas can indeed work in parallel most of the time. In the final implementation all the replicas are enabled at once; because of the low granularity of moves in the MKP solver, a replica can use a neighbor's result from the previous MCS, accepting a possible error of at most 2 bits. Simulations confirm that the quality of result is equivalent to the one with the partitioned replicas. By allowing all replicas to process at the same time, overall latency is improved by about 50%.

3.2.2 Controller

The Controller module fulfills multiple functions. Its main role is to keep replicas synchronized over the same Monte Carlo step, by issuing the `outEnable` signal to

all processors and releasing it once it receives the `doneEO` signal from all replicas for that step. It also handles the transfer from memory of the 16-bit values for J_t. A typical QA run can take a few million simulation steps in order to reach a good quality solution. Since the calculation of J_t involves complex trigonometric functions, and the same values can be reused in every simulation run, it is more reasonable to precompute all values for J_t and store them in a fast on-chip memory outside the solver module.

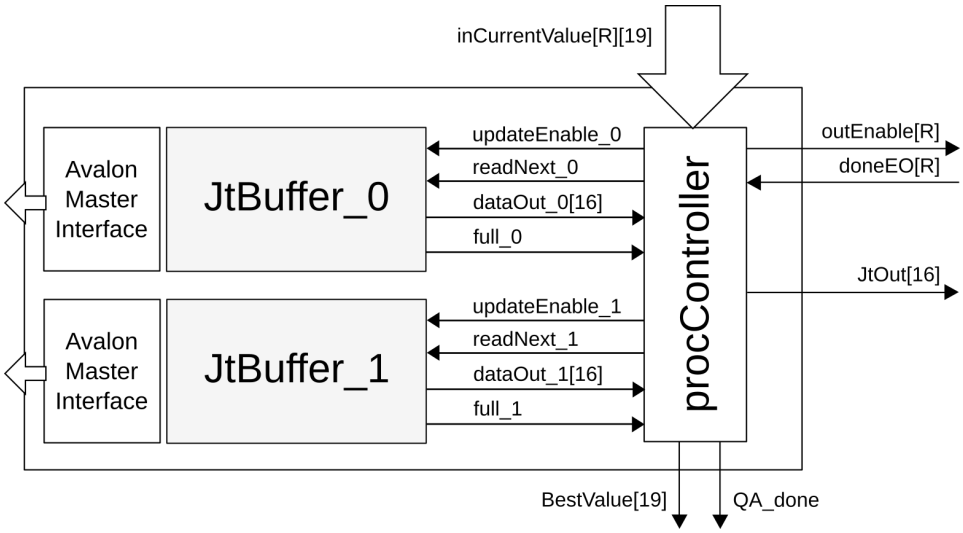


Figure 3.7: Block diagram of the Controller module

The memory is a RAM connected to the Avalon bus with a Memory Mapped Slave interface [42]. The corresponding Master interface is implemented within two `JtBuffer` modules inside the Controller. When the solver is reset, the Controller enables the first buffer, which issues a burst read request on the Avalon bus. After receiving the first 50 values, the buffer is full, and it raises the `full` signal; the Controller then disables it and enables the second buffer who independently begins to fetch the following 50 values. At the same time, the Controller raises the `readNext` signal to request the first J_t value from the first buffer; as soon as it is available the Controller forwards it to the replicas and enables the first annealing step. When all the replicas are done updating, the controller disables them and requests the next J_t value from the buffer. Then the replicas are enabled and a new MCS begins. When all J_t values in a buffer have been consumed, it deasserts the `full` signal; the Controller then enables the update for that buffer and switches to reading from the other one, as long as it is full. Since the burst transfer takes less time than the execution of 50 simulation steps, there is generally no stall in switching from a buffer to the other.

The Controller also receives the fitness of each replica's current solution and determines when a new optimum has been found, outputting a new `bestValue`.

Once all simulation steps have been executed, the Controller raises the `QA_done` output signal and stops the annealing.

3.2.3 RQA Engine

The Restricted Quantum Annealing (RQA) engine's role is to keep track of the frequency of appearance of each item across solutions. I accomplished this by means of a SWAR algorithm for popcount (or Hamming weight counter), which is essentially a series of $\log(R) + 1$ sum, right shift and bit masking steps.

For each item i, the input to the popcount is built out of a vertical slice of all the replicas' item vectors, taking from each only the ith bit (as highlighted in red in Fig. 3.8). The result of the popcount is stored in a *frequency vector* at position i. If the frequency is greater than the RQA *blocking frequency*, the ith bit of the output signal `lockedItems` is set.

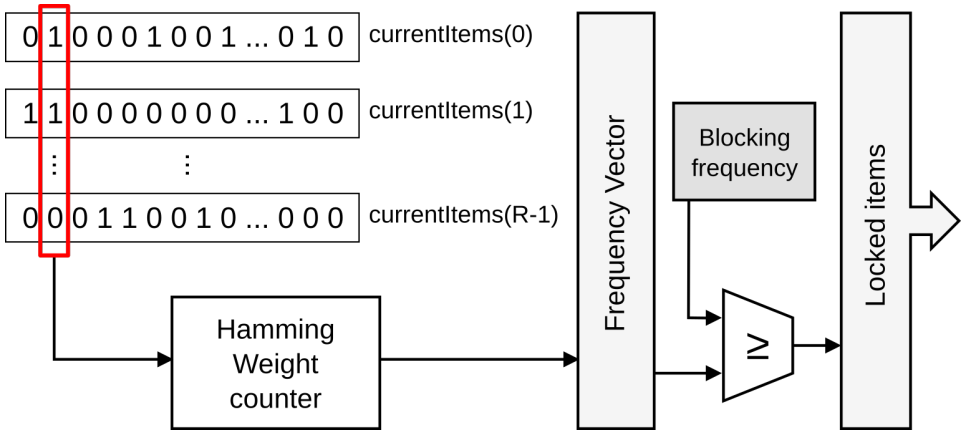


Figure 3.8: Block diagram of the RQA engine

Using popcount lets us avoid computing long sums with the 500-position item vectors of the hardest benchmark. Every vertical slice built is R bits long, which is generally much shorter than the item vectors.

Additionally, it is possible to explore the design space for this component by performing loop unrolling to control the quantity of items processed at the same time and the pipeline depth of the frequency counter. I found that the smallest-area, smallest-delay, longest-latency implementation is the most efficient: one item is processed at a time, and CWB's automatic scheduling reduces the delay as much as possible, resulting in a $(\log(R) + 1)$-cycle latency.

Although the replicas don't have access to the most updated version of the `lockedItems` vector at all times, RTL simulation results show that this minimized implementation has no adverse effects on the speed of convergence of RQA or the quality of result. Then, RQA can be added to the system with negligible impact on area and maximum frequency.

Chapter 4

Results and discussion

4.1 Software testing results

4.1.1 Ising spin glass model

At first, I performed a feasibility study for the Ising spin glass model. I began by trying to reproduce literature results about the 2d lattice, then moved on to increasing connectivity modes. The final goal was to test whether the fully-connected Ising model could be implemented with Quantum Annealing using a parallel hardware architecture based on minor embedding, similar to the D-Wave machine's implementation.

The 2-d spin glass model

We performed a comparison of the performance of QA and SA on the 2d spin glass model. QA is run with $R = 50$ replicas, the transverse field strength is set to an initial $\Gamma_0 = 3.0$ and decreased to 0 with a linear schedule. The algorithm performs both local and global moves in the way described in section 3.1.1. SA is performed with an initial temperature of $T = 3.0$, also decreased to 0 with a linear schedule. Both algorithms were implemented in C++. The benchmark for the test is a 32×32 spin lattice with randomly generated couplings $[J_r]$ which are real numbers uniformly distributed between -2 and +2. We performed simulation of this model for different values of the annealing time τ, up to 3 million steps.

The results of this simple test confirm those found in literature. Figure 4.1 shows the final energy per spin of solutions reported by QA and SA, depending on the number of simulation steps; the lower the energy value, the closer the solution is to the exact solution (ground state of the system). Each data point is averaged over the results of 10 trials. The dashed line `QA-eq` shows the results of QA adjusted for equivalent computation effort to SA, that is, the corresponding data point was actually computed in $\frac{\tau}{R}$ simulations steps, to take into account the added

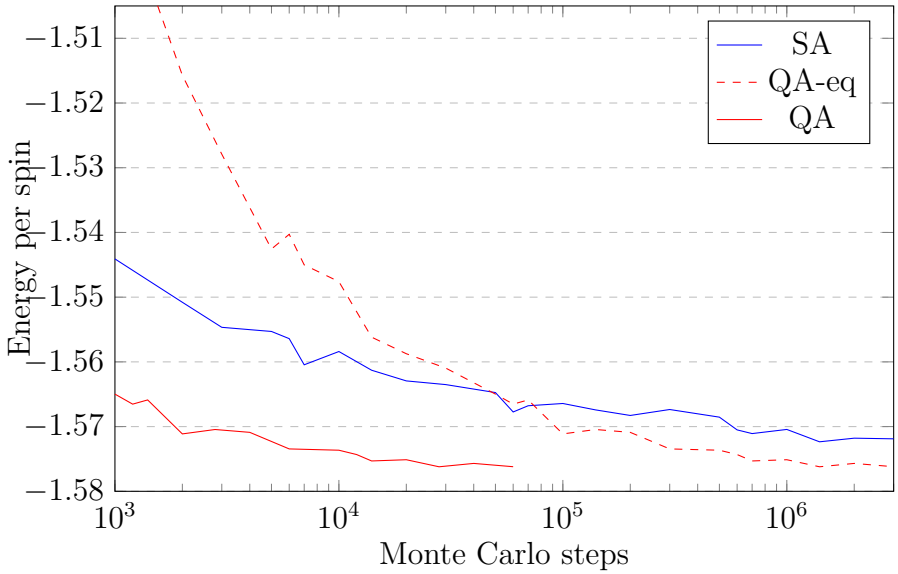


Figure 4.1: Performance comparison of SA and QA on the 2d Ising model

computation load due to the multiple replicas. Even when considering equivalent computation effort, QA's performance surpasses SA's at a crossing point around $\tau = 100{,}000$. When taking into account the parallelization of replicas (continuous red line), QA is able to converge to a better solutions in a lower number of Monte Carlo steps than SA. We can conclude that the 2-d spin glass model is one where the application of QA is beneficial.

For the sake of hardware acceleration, existing FPGA realizations of 2d spin glasses could be easily extended to the QA model, at the cost of larger area and more complex connectivity. New models of FPGA with large logic availability would allow this.

I decided not to pursue hardware acceleration of the 2d spin glass model for two reasons. Unlike higher order models, there already exist numerical methods that solve the 2d spin glass problem exactly. This makes QA of the 2d spin glass model unappealing as a simulation of a physical system. Moreover, I didn't find any evidence that any NP optimization problem can be mapped to 2d efficiently, which means it would not be useful as a general-purpose heuristic solver.

The 3-d spin glass model

We ran another simple test to verify the performance of QA on the 3d model. The algoithm parameters used are the same as those listed above for the 2d model. The benchmark for the test is a $10 \times 10 \times 10$ lattice with $[J_r]$ uniformly distributed between -2 and +2. Because of the added complexity of the graph, execution times

are much longer than for the 2d model, so I was able to perform simulation for a limited number of steps (up to $\tau = 30{,}000$). The results of this partial test are reported in figure 4.2.

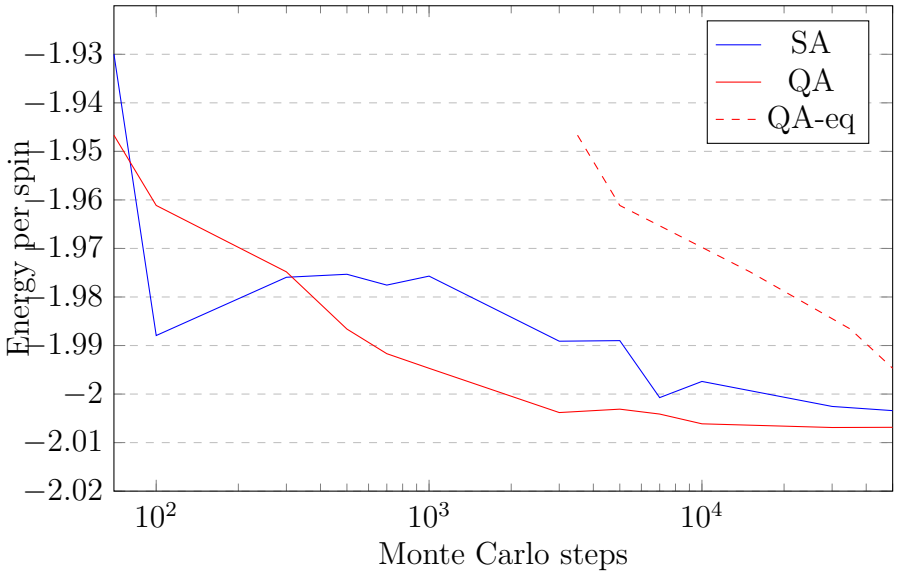


Figure 4.2: Performance comparison of SA and QA on the 3d Ising model

QA still delivers a better performance than SA in terms of number of MCS, but QA showed worse scaling on this model, and its advantage over Simulated Annealing is not as strong as for the 2d model.

Minor embedding of the fully connected model

The fully connected model does not allow parallel spin updates and therefore did not seem like a good candidate for parallelization. To work around this problem I decided to test the minor embedding technique with the Chimera graph. In order to test the effectiveness of the mapping, first I prepared a simple problem on a fully connected graph. The problem has $N = 80$ spins and J_r couplings uniformly distributed between -2 and +2. I verified that both QA and SA are able to find the ground state of this problem in few thousand steps on the fully connected Ising model. I then proceeded to map the fully connected graph to a 20×20 triangular matrix of Chimera subgraphs by the iterative mapping method. The parameters for SA and QA are the same as described above, with the addition of the ferromagnetic coupling $J_F = \sqrt{N}$ among qubits belonging to the same Logical Bit group.

As it is evident from figure 4.3, the results of this test were fairly discouraging. Running SA on the embedded fully-connected Ising model takes a higher number of simulation steps than on the non-embedded version, but it eventually converges

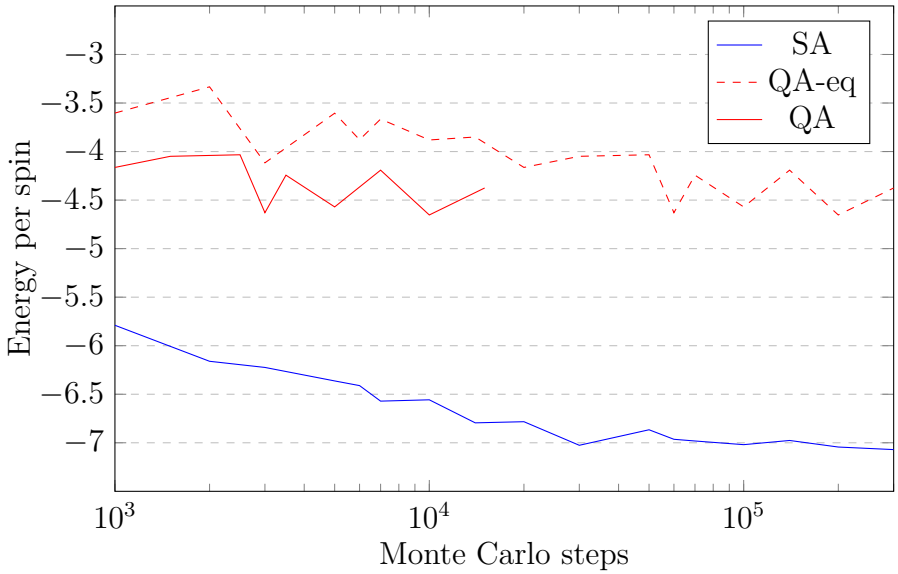


Figure 4.3: Performance comparison of SA and QA on the fully connected Ising model with Chimera

to a solution within few percent of the ground state, confirming the effectiveness of the Chimera mapping.

However, SQA cannot be applied to the embedded fully-connected Ising model in a straightforward way. The ferromagnetic couplings J_F among qubits representing the same spin compete with the coupling J_t among imaginary-time replicas, increasing the frustration of the system. In essence, the single spin dynamics are frozen and the freedom of exploration in the solution space is very low. Attempts to reduce the value of J_F result in a more disordered system that often returns invalid solutions where spins in a single qubit are not fully aligned.

Venturelli et al. [31] suggested it's possible to overcome the freezing by using cluster updates; however, adding a complex cluster-building algorithm would defeat the purpose of trying to simplify the architecture by minor embedding. In addition, parallelization of clustered version would be even more difficult. As such, the realization of a general-purpose QA solver based on fully-connected Ising with spin-flip dynamics is not possible without significant modifications to the algorithm.

4.1.2 Multidimensional Knapsack Problem

Given the limitations found in the Ising model, I decided to explore a different implementation with the definition of a problem-specific algorithm, where Quantum Monte Carlo dynamics can be applied without the rigid spin glass structure. I wrote three software solvers for the Multidimensional Knapsack Problem (MKP) in C++ for Simulated Annealing, Quantum Annealing, and the Hybrid PSO-Genetic

Optimization algorithm (HPSOGO); the QA solver is designed to optionally apply Restricted Quantum Annealing to improve the results. The implementation of the three algorithms is explained in detail in section 3.1.2.

Testing architecture

I ran an exhaustive test for all three algorithms on two well-known benchmarks for the MKP, the SAC-94 [43] and the ORLib [49] problem sets. Simulated Annealing ran for τ = 1,000,000 steps with initial temperature T_0 = 3,000 and a linear annealing schedule. Quantum Annealing results are reported here for τ = 1,000,000. The number of replicas is set to $R = 32$ and the initial transverse field parameter is $\Gamma_0 = 3.0$ with a linear annealing schedule. HPSOGO instantiates 15 replicas and runs for 10,000 iterations; empirical tests showed that, on the average, these parameters gave similar computation times to QA of a million steps. The social and cognitive mutation weights are set to $c_c = c_s = 2.05$. I also tuned the velocity calculation by multiplying the result of equation 2.3 by a factor $\chi = 0.729$.

Tests were repeated 20 times per benchmark and the average quality of solution is reported in detail in appendix A, together with the elapsed CPU time. The parameters evaluated in the test are:

- **Success Rate (SR)**: because HPSOGO has a chance of returning invalid solution, it is necessary to evaluate this parameter. The SR is calculated as the ratio of valid solutions returned over the total number of algorithm runs and is expressed as a fraction. Because of the modifications I applied to the HPSOGO penalty function, SR for all the algorithm runs in the test is 1.
- **Mean Absolute Percent Error (MAPE)**: a percent measure of the mean distance of solutions from the optimum. Given the known optimal solution for the instance S_{opt} and solution S_i returned by the algorithm at iteration i, the MAPE over K iterations is:

$$\frac{\sum_{i=1}^{K}(S_{opt} - S_i)}{K} \cdot \frac{1}{S_{opt}} \tag{4.1}$$

- **Least Error (LE)**: the error of the best solution returned by the algorithm.
- **Standard Deviation (SD)**: the SD of solutions returned by the algorithm. It is a measure of the reliability of the algorithm in always producing the same quality of result.
- **CPU time**: the computation time of the program in seconds, evaluated using the `clock()` function from ISO C to measure the number of cycles and divide the result by the system-dependent macro `CLOCKS_PER_SEC`. All our implementations are single-threaded and they ran on an Intel i7 CPU at 2.67 GHz.

Test results (SAC-94)

SAC-94 is a library of benchmarks compiled from several sources in the literature. It is a small set of 40 instances, but it contains a variety of parameters and most of the instances are based on real-world problems. The average quality of solution is reported in the graph, grouped by benchmark family and number of constraints:

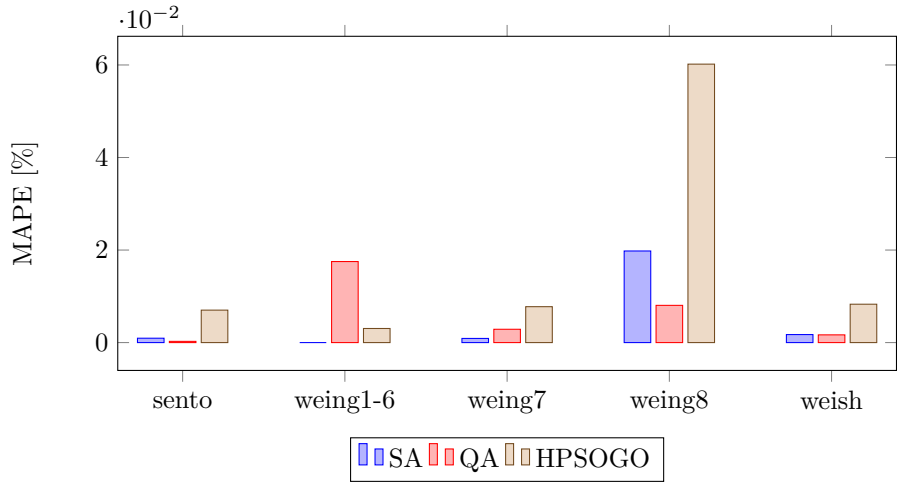


Figure 4.4: Performance comparison of SA, QA and HPSOGO on the SAC-94 library

I have isolated the results for benchmarks `weing7` and `weing8` as the results showed great disparity to the rest of the library. As we can see from the bar graph in 4.4, Simulated Annealing had the most success in finding good quality solutions across the library.

The Weingartner benchmarks 1 through 6 proved to be especially difficult for Quantum Annealing. It is difficult to determine an exact reason for this because the original paper [44] does not give details about how these benchmarks were generated other than the fact that they are based on real-life capital budgeting problems. Looking at the problem instances I conjectured that the poor performance of QA may be due to the fact that these problems, unlike the others in the library, assign a very wide range of values to the item profits ($100 \sim 30{,}000$); this might make QA reluctant to get rid of high-profit items that are not actually present in the optimal solution. If this is the case, it will suffice to tune the QA parameters to this particular instance to obtain better results.

While HPSOGO gave better results for Weingartner 1 through 6, QA outperformed it on `weing7` and `weing8`, and did even better than SA on `weing8`. These two instances have a very high number of constraints, 105, in contrast to `weing[1-6]` which have only 28 constraints. This confirms the fact that QA does better in cases where the problem is highly constrained.

QA also did better than SA on the `sento` benchmarks, and matched its performance on the `weish` benchmarks.

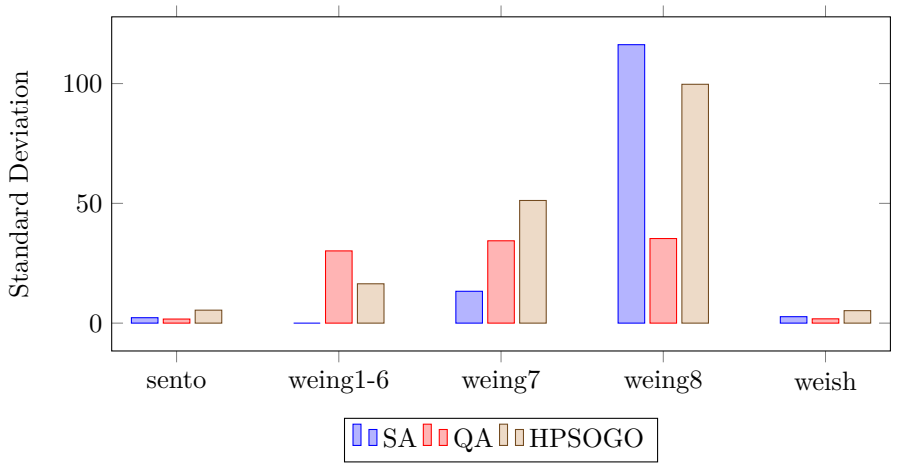


Figure 4.5: Standard Deviation for SA, QA and HPSOGO results on the SAC-94 library

Chart 4.5 shows the standard deviation for the three algorithms. Once again, SA had the lowest deviation overall, and the Weingartner family of instances proved the most difficult, with `weing8` causing even SA to produce wildly different results.

It is interesting to note that the standard deviation for QA remained the same throughout the Weingartner benchmark without being impacted by the number of constraints.

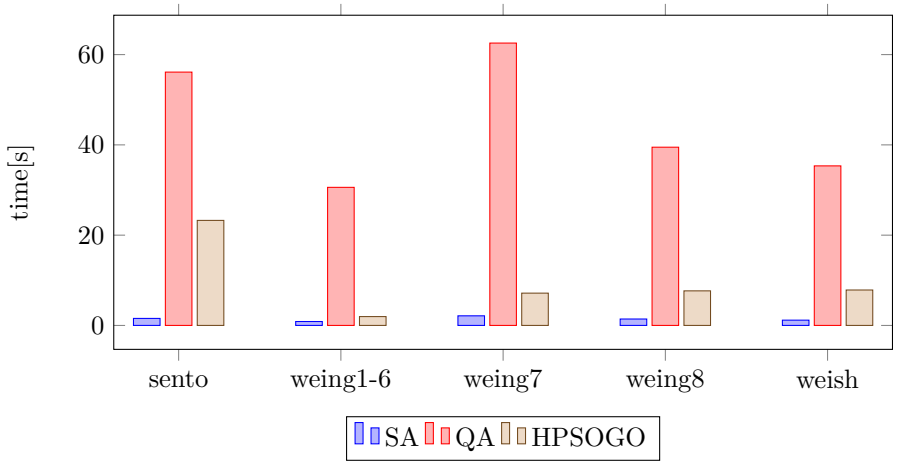


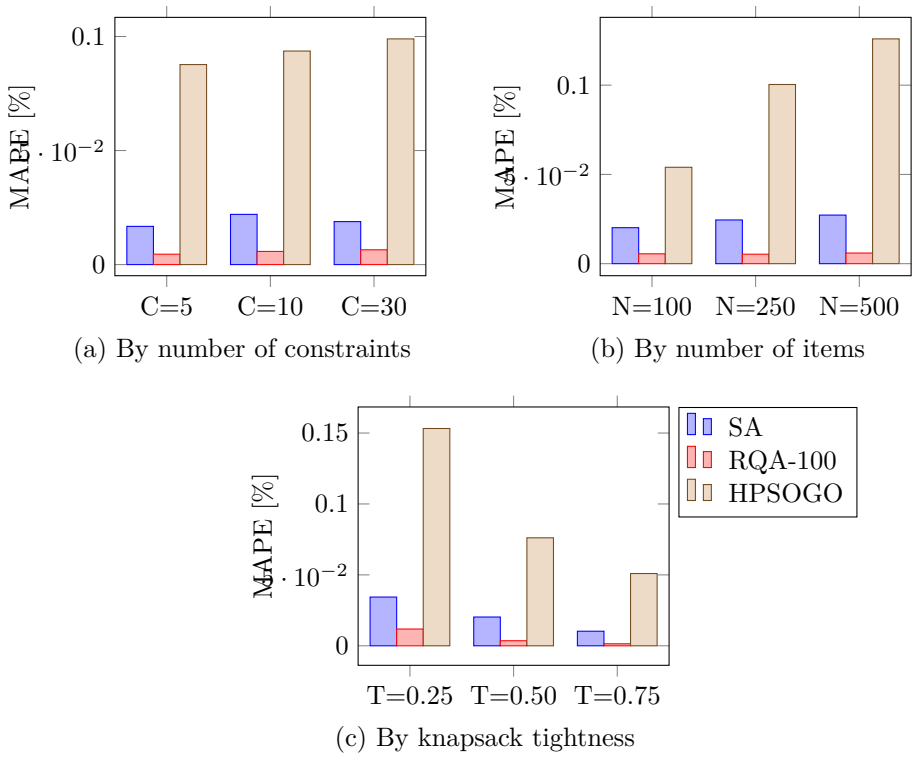
Figure 4.6: Computation times for SA, QA and HPSOGO results on the SAC-94 library

As for the computation times, as expected, SA is the fastest, and QA takes

much longer than other methods due to the added complexity of the Trotter replicas (however, appendix A include the results for QA of 100,000 steps, which show that QA can converge to a solution faster for a few of these benchmarks). It is worth noting that HPSOGO takes a particularly long time on the `sento` instances. This may be because these problems have a larger size (30 items, 60 constraints), which complicates the calculation of the penalty function within this algorithm. In general, the HPSOGO computation time scales badly with the number of constraints, while SA and QA are not significantly affected by it.

Test results (ORLib)

The ORLib benchmark was formulated in [45] for solution via Genetic Algorithm. It is a set of 270 MKP instances with different sizes. The problems in this benchmark are characterized by their tightness, a measure of capacity of the knapsack; instances with a tightness of 0.75 are less tight (constraints are more relaxed, more items can be inserted) and ones with a tightness of 0.25 are tighter. Since this benchmark contains such a large number of instances, the results are reported here aggregated by problem dimension.



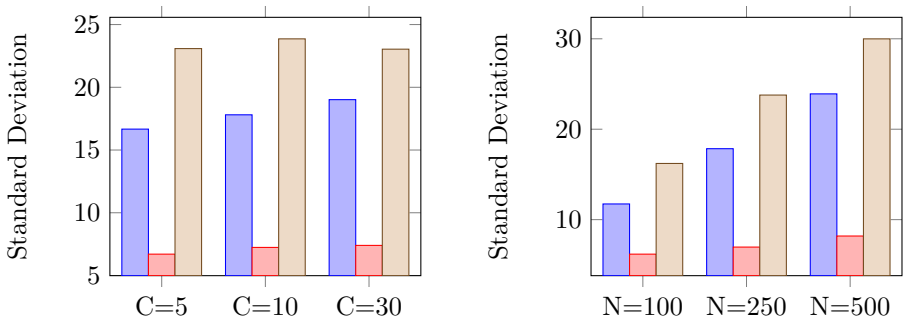
(a) By number of constraints

(b) By number of items

(c) By knapsack tightness

Figure 4.7: Performance of SA, QA and HPSOGO on the ORLib library

The best performing version of QA on this benchmark is Restricted Quantum Annealing with a blocking frequency of 100%. The results in chart 4.7 show a definite dominance of QA over the other two methods when it comes to solution quality. All three algorithms are slightly affected by changes in the number of constraints, while the number of items affects quality of results more significantly, especially for HPSOGO.



(a) By number of constraints

(b) By number of items

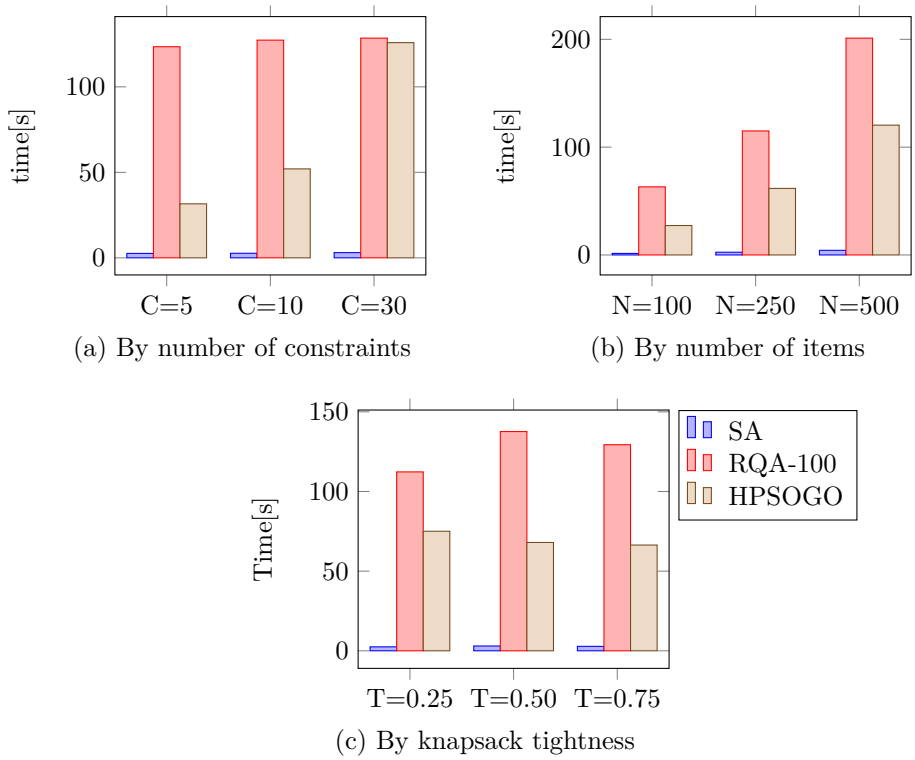
(c) By knapsack tightness

Figure 4.8: Standard deviation for SA, QA and HPSOGO results on the ORLib library

Chart 4.8 shows the standard deviation. QA results also report the lowest standard deviation among solutions. Interestingly, while all three algorithms have worse SD when the number of items grows, only QA seems to be significantly influenced by the knapsack tightness.

Finally, timing results are reported in chart 4.9. SA remains the fastest algorithm with an average computation time of few seconds. QA's CPU-time lengthens as the number of items grows, but remains constant with the number of constraints. HPSOGO's running time depends both on the number of items and on the number of constraints; the average elapsed time for 30 constraints is on par with QA.

Overall, I found a marked difference of performance results among benchmark tests depending on the size of the problem instance and the instance parameters. On



(a) By number of constraints

(b) By number of items

(c) By knapsack tightness

Figure 4.9: Computation time for SA, QA and HPSOGO results on the ORLib library

smaller instances, such as those included in the SAC-94 benchmark, SA outperforms the other algorithms both in speed of execution and in quality of result.

However, even among small instances there exist some problems, such as those in the Weingartner family, that give surprising results; QA finds extraordinary difficulty in solving some of these while it outperforms the other algorithms in others; this can probably be attributed by the instances having non-uniformly distributed parameters such that some strategies are more effective than others in exploring their solution space.

My version of HPSOGO showed worse results on SAC-94 than those reported in [40], probably due to the adjusted penalty function. However, it still outperforms QA on the smaller Weingartner problems. I also believe that the timing measurement is still indicative of the computation effort required for the algorithm.

As for the much larger instances in the ORLib benchmark, Restricted Quantum Annealing reported much better results, with an overall greater quality of result and reliability, especially for the bigger problem instances, at the cost of longer computation times. Therefore, it seems well worth it to pursue hardware acceleration of the MKP-RQA solver in order to tackle these difficult instances.

4.2 FPGA synthesis results

I wrote the description of the hardware implementation of the MKP-RQA solver (described in section 3.2) behaviorally in SystemC and performed high-level synthesis with the *NEC CyberWorkBench HLS compiler* [41] (CWB), then I exported the resulting Verilog files and executed logic synthesis with Altera Quartus Prime after adding to the design an Avalon-MM bus and a fast on-chip RAM for storing the J_t entries. The target is an Altera Stratix V [46] FPGA board.

4.2.1 Logic synthesis results

In table 4.1 are reported the logic synthesis results for RQA-MKP solver alone. Parameters are $R = 16$, $\tau = 1{,}000{,}000$, 500 items and 30 constraints; The OR30x500 family of benchmarks is the biggest one available in literature and the one I considered for final implementation.

Pre-placement area	
Device Family	Stratix V
Logic utilization (in ALMs)	N/A
Total registers	143108
Total pins	2
Total virtual pins	168
Total block memory bits	2,457,600
Post-placement area	
Family	Stratix V
Device	5SGXEA7N2F45C2
Logic utilization (in ALMs)	147,236 / 234,720 (63 %)
Total registers	157782
Total pins	2 / 1,064 ($<$ 1 %)
Total virtual pins	168
Total block memory bits	2,457,600 / 52,428,800 (5 %)
Total RAM Blocks	128 / 2,560 (5 %)
Critical path	
Worst case maximum frequency f_{max}	164 MHz
Power consumption	
Total Thermal Power Dissipation	7133.23 mW
Core Dynamic Thermal Power Dissipation	5875.28 mW
Core Static Thermal Power Dissipation	1235.37 mW
I/O Thermal Power Dissipation	22.58 mW

Table 4.1: Results of logic synthesis for the MKP-RQA module

After a first synthesis of the RQA-MKP module, I connected the design to the Avalon-Memory Mapped bus (as shown in Fig. 3.4) and added the fast on-chip RAM of 1,000,000 16-bit entries that contains the J_t values. The result of logic synthesis for the whole architecture are reported in table 4.2. Logic synthesis was performed in Aggressive Performance Optimization Mode, which means that area and power savings were sacrificed to meet timing constraints. Register duplication was turned on for physical synthesis and routing to further improve the timing performance.

The addition of the bus and external memory increases the power consumption, but the final implementation also improves on the critical path with a higher worst-case f_{max}. The high usage of block memory bits is due to the fact that every processor stores a local copy of the item weights ($30 \times 500 = 15{,}000$ entries of 18 bits for 16 replicas): this is to improve reading times and avoid access conflicts, since every processor in the module should be allowed fast random access to any row of values at any time to check for constraint violations at the item insertion step.

4.2.2 Result analysis

Area

Since the problem size and parameters are hard-coded into the implementation, I performed synthesis multiple times for a variety of problem sizes and number of replicas. Figure 4.10 shows the estimated area for realizations of a `OR5x100` benchmark solver. The total area of logic utilization grows linearly with the number of replicas R; this is to be expected since every replica added corresponds to a new processor core in the architecture.

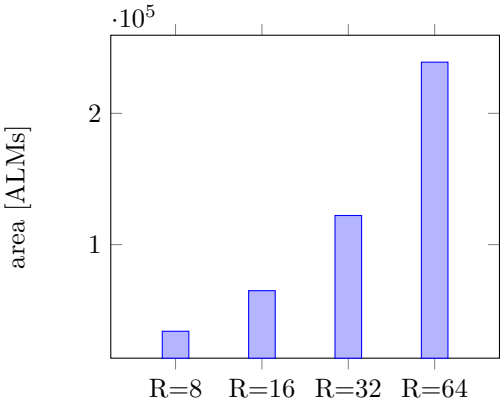


Figure 4.10: Area estimation as a function of R

Examining the growth of area when varying the number of constraints and items (figure 4.11) shows that area also grows linearly with the problem size. The reason

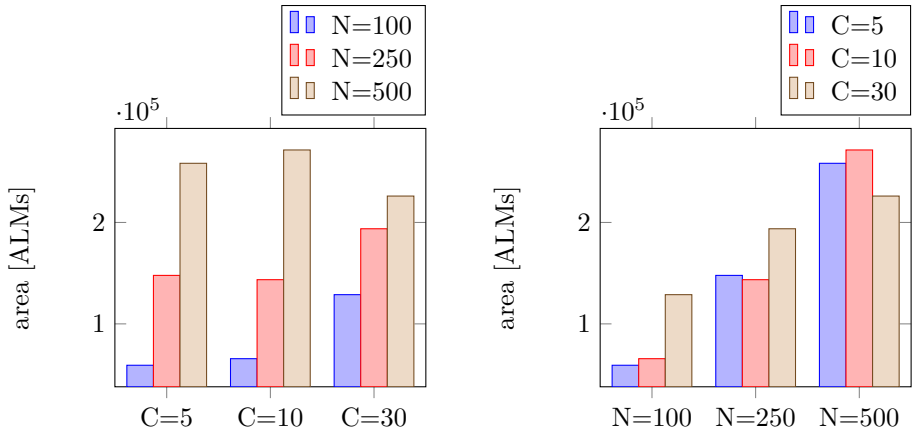
Pre-placement area	
Device Family	Stratix V
Logic utilization (in ALMs)	N/A
Total registers	143262
Total pins	86
Total virtual pins	0
Total block memory bits	18,457,600
Post-placement area	
Family	Stratix V
Device	5SGXEA7N2F45C2
Logic utilization (in ALMs)	148,145 / 234,720 (63 %)
Total registers	160936
Total pins	86 / 1,064 (< 8 %)
Total virtual pins	0
Total block memory bits	18,457,600 / 52,428,800 (35 %)
Total RAM Blocks	1,105 / 2,560 (43 %)
Critical path	
Worst case maximum frequency f_{max}	178 MHz
Power consumption	
Total Thermal Power Dissipation	8559.14 mW
Core Dynamic Thermal Power Dissipation	7110.49 mW
Core Static Thermal Power Dissipation	1381.29 mW
I/O Thermal Power Dissipation	67.36 mW

Table 4.2: Results of logic synthesis for the entire architecture

the area decreases only for the largest instance ($C = 30$, $N = 500$) is because beyond a certain size threshold the CWB synthesis engine automatically maps the memory containing the item weights to block memory instead of LUT-RAM. The implementation of the weights memory is defined by a high-level macro provided by the CWB software; due to time restrictions, I was not able to re-perform synthesis by forcing the synthesis engine to utilize block memory for the smaller instances, but it can be achieved by changing the aforementioned threshold in CWB's synthesis settings. This reduces logic utilization while introducing some overhead in the RAM access time.

Critical path

I compared the maximum frequency estimated by Quartus Prime over a wide range of synthesis results corresponding to various combinations of number of replicas,



(a) By number of constraints (b) By number of items

Figure 4.11: Area estimation as a function of C and N

number of items, and number of constraints.

From Fig. 4.12, it is evident that the worst-case maximum frequency is decreasing as the area grows. There are at least two main reasons for this: first, combinational paths become longer and slower as the width of parameters grows, especially of the item vector. Second, interconnections also become longer and more complex, causing increased delays. In practice, increasing the number of replicas or the size of the problem makes the system slower.

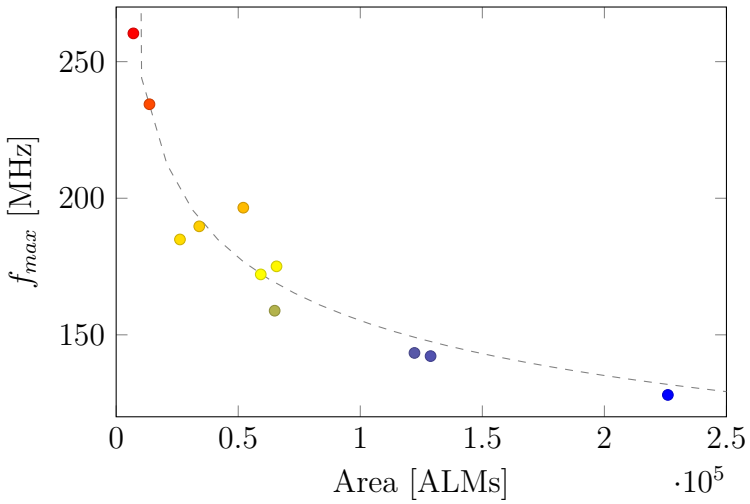


Figure 4.12: Maximum frequency obtained by Quartus synthesis for varying hardware size.

Latency

I created a bit-compatible integer model in software to estimate performance of an FPGA version ahead of implementation and verified that its behavior matches the floating-point version exactly. Then, it is fair to compare the behavior of the FPGA implementation (in RTL simulation) with the floating-point software version of the algorithm. Fig. 4.13 shows the results of the three versions for the benchmark OR30x500-0.25, $R = 16$. The behavior is completely coherent in the two versions, displaying similar convergence across a wide range of τ; quality of result is not lost in the transition from floating point (fp) to fixed point data representation.

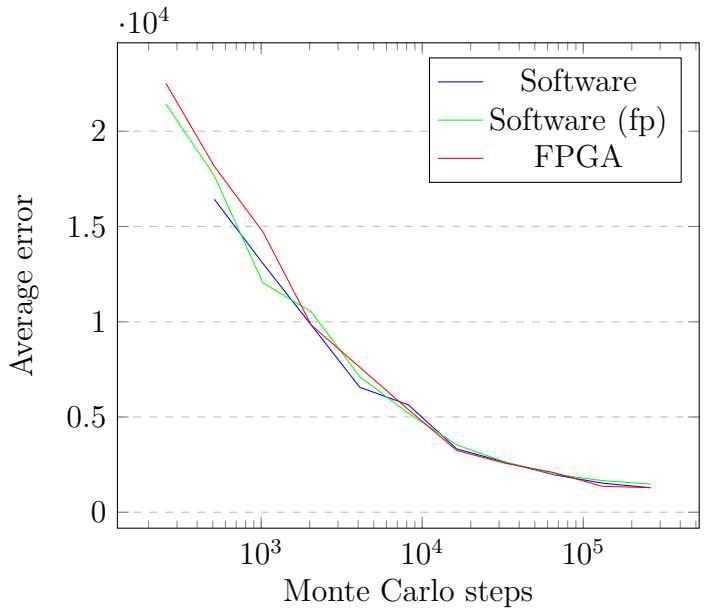


Figure 4.13: Comparison of results from the FPGA and software versions of the algorithm

I used the integer model software to estimate the number of cycles needed for the FPGA to execute longer simulations. In the integer version, the number of cycles per step increases quite strongly as τ increases, moreso than in the software version; this is probably due to non-idealities in the random number generation using LFSR. Since the number of items is not necessarily a power of two, some of the random numbers must be discarded, wasting a cycle, and the likelihood of this happening increases with the length of the simulation. I compared the average latency per step of the FPGA implementation with the average number of RNG trials in the software version. It can be assumed that the two parameters are directly proportional as the FPGA latency per step is predominantly determined by the number of LFSR trials, with little (and generally constant) overhead from the rest of the algorithm. Fig. 4.14 shows how the cycle latency due to the RNG

trials, after a sharp initial growth, settles into a slowly increasing curve, compatible with that encountered in the software simulation. This means that, past a certain threshold (around 100,000 steps), the FPGA implementation's speedup will likely maintain its advantage over the software version even as we increase the annealing time.

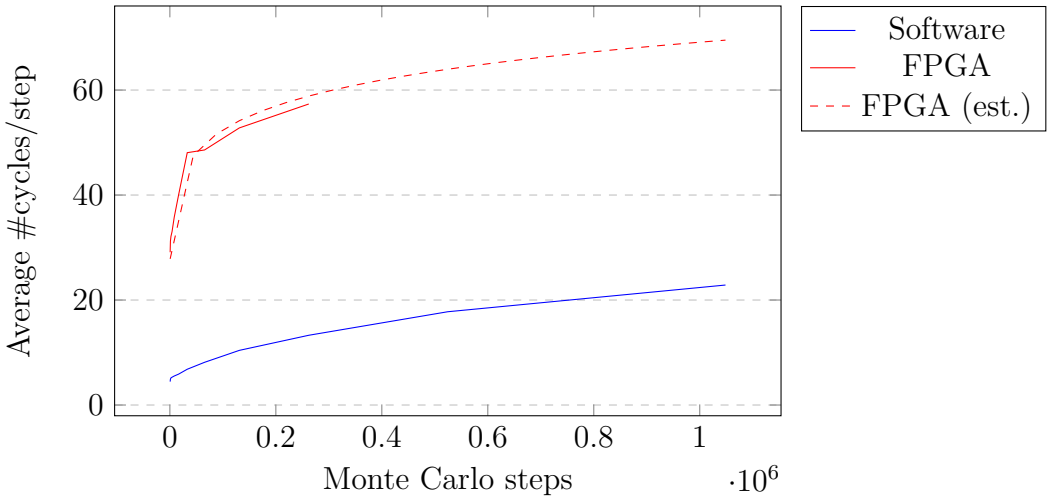


Figure 4.14: Comparison of random trials per step as a function of τ, benchmark problem OR30x500-0.25. The regression function used for the FPGA estimation is $f(\lambda) \approx 13.17\lambda^{0.12}$.

Knowing the average number of cycles per step c_{avg}, I estimated the execution time for different values of τ as $\tau \cdot \frac{c_{avg}}{f_{max}}$. The execution time for $f_{max} = 164\,\text{MHz}$ is charted in Fig. 4.15 against the CPU-time measured for the integer-point software version. The FPGA implementation is nearly 150× faster than the software at executing the same number of Monte Carlo steps. The execution time of our solver even compares well to the results of the exact branch-and-bound GPU solver from [47], which reports a time of 1.44 s to solve a 500-item problem with only one constraint; the FPGA implementation can execute a million simulation steps of a 500-item problem with 30 constraints in 421 ms.

Finally, Fig. 4.16 shows the speed of convergence to result for the two versions of the algorithm. It is clear that the hardware version can produce a similar quality of result as the software version in less computation time.

Power

Though synthesis provides limited information on the power consumption, it is still possible to make an optimistic estimation of the energy savings.

The software version ran on a computer with an Intel i7 920 CPU; the datasheet [48] reports a maximum I_{cc} per core equal to 145 A and a typical associated V_{cc}

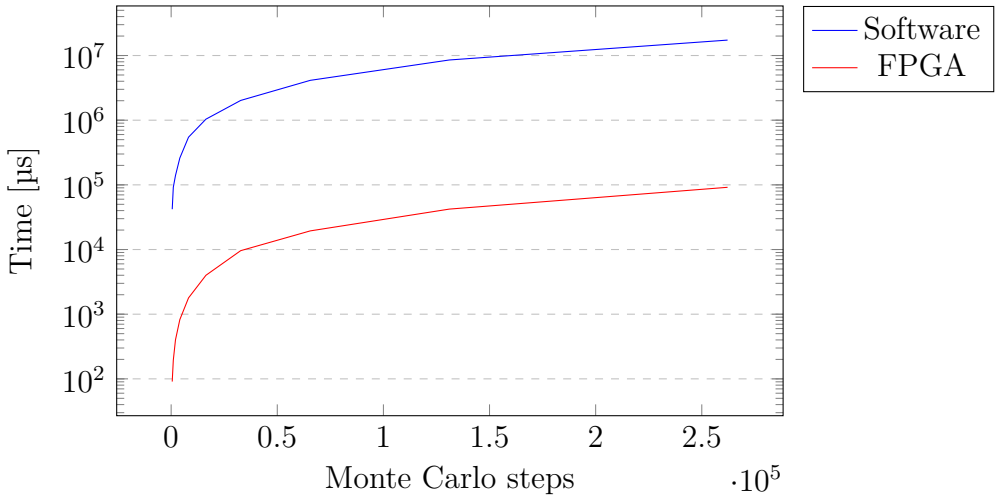


Figure 4.15: Execution times for the FPGA and software versions for the benchmark OR30x500-0.25

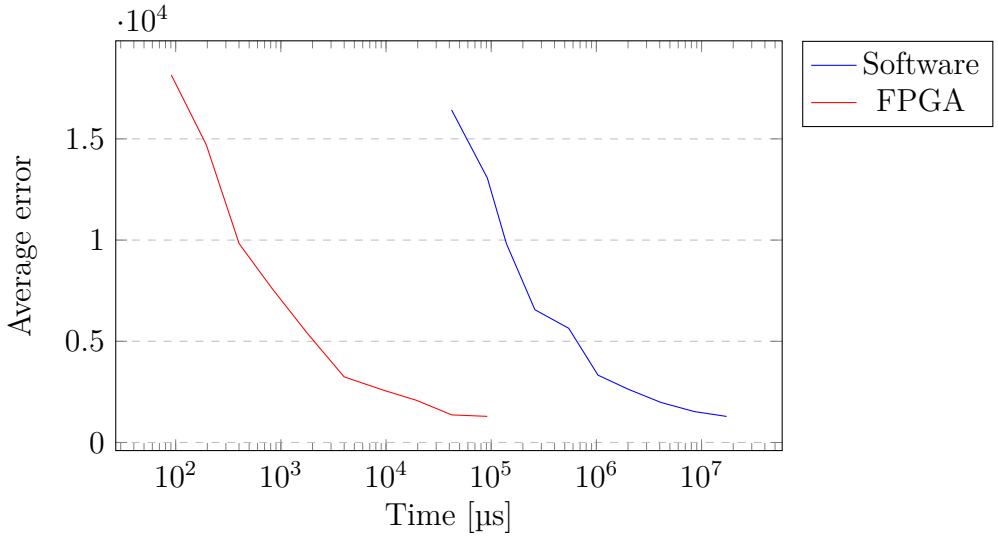


Figure 4.16: Speed of convergence to result for the software and FPGA versions, benchmark problem OR30x500-0.25

per core of 0.131 V. Assuming our single-thread software ran on a single core at full load, that would mean an instantaneous power of 18.995 W; since running QA for 250,000 steps takes 17.319 s, the estimated energy consumption is about 329 J. Meanwhile, the FPGA energy consumption for 250,000 steps is around 8.559 W $\times$ 0.108 s = 0.924 J, leading to estimated energy savings of 99.7%.

Chapter 5

Conclusion

In order to determine the implementation method for a hardware Quantum Annealing solver, a feasibility study was performed for the fully-connected Ising model, targeting the Chimera graph as a candidate for hardware implementation. Despite this being a well-tested model for Adiabatic Quantum Computation, test results showed that the Chimera graph is in fact not a favorable implementation for Simulated Quantum Annealing, because of inherent limitations in this classical algorithm.

Extensive tests were ran on the Multidimensional Knapsack Problem, solving every instance in the SAC-94 [43] and ORLib [49] benchmarks. We compared solution quality of Quantum Annealing with Simulated Annealing and a Hybrid PSO-Genetic algorithm [40]. Results confirmed that, while Simulated Annealing delivers the best solution quality for small instances, QA outperforms other methods more and more significantly as the size of the problem increases, delivering highly optimized results with a low variance across algorithm iterations. The cost of Quantum Annealing is a much longer execution time, which however is still reasonable (within few minutes) when compared to exact solvers like CORAL [13] which report execution times of several hours for these instance sizes. Therefore, application of hardware acceleration for QA of large MKP instances seems beneficial.

The hardware architecture was written in SystemC language and synthesized with a Stratix V FPGA [46] as target; the parametric design instantiates multiple computation cores, synchronized by a Controller module that regulates the annealing process. An optional module was implemented to further optimize computation by applying Restricted Quantum Annealing [32], allowing to improve the quality of result with a negligible cost in terms of area and latency. Since large problem instances require long simulation times, precomputed data utilized during the simulation is stored in a fast on-chip RAM, connected to our design via the Avalon-MM bus [42]. Burst access to this memory through two alternating buffers allows to access the data with a negligible latency overhead.

RTL simulation proves that our QA solver provides the same quality of result as the floating point software version. The final implementation, tailored to the largest-size instance in the ORLib benchmark, utilizes 63% of the total area on the target board, with a reported maximum frequency of 164 MHz. Analysis of the average algorithm latency shows that the QA solver is about 150× faster than the software implementation on a general-purpose CPU. The power consumption also compares favorably to the software implementation, with estimated energy savings of 99.7%.

All in all, parallelization of the Quantum Annealing algorithm can be successful because of the population-based nature of the algorithm. The fact that the quality of solution is comparable to the software version shows that the quantum tunneling emulation properties of path-integral QA [26] are preserved despite the approximations of the fixed-point model.

The hardware described in this thesis can be modified to run optimization problems different from MKP, by rewriting the behavioral description of the processor module to fit the new problem; however further study would be required in order to determine a feasible data structure for the implementation of the solver, as well as storage of the benchmark values describing the instance. The main difficulty of the implementation lies with the high volume of this data, which must be available for fast random access to all parallel processors at any time during the simulation. While this architecture achieved it by storing local copies of all data for each processor, future implementations should strive to optimize storage of the problem data and devise suitable access schemes in order to reduce block memory usage and accommodate even larger instances.

Appendix A

MKP Benchmark Results

This appendix reports detailed results for the Simulated Annealing (SA), Hybrid Particle Swarm/Genetic Optimization (HPSOGO) and Quantum Annealing (QA) algorithms on the SAC-94 and ORLib benchmark instances for the Multidimensional Knapsack Problem.

The evaluated parameters are Success Rate (SR), Mean Absolute Percent Error (MAPE), Least Error (LE), Standard Deviation (SD) and computation time in seconds. The definition of these parameters can be found in section 4.1.2.

A.1 SAC-94 benchmark

Quantum Annealing results are reported for τ = 100,000 (QA-100k) and τ = 1,000,000 (QA-1M).

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
sento1	30	60	HPSOGO	1	0.0090	11	5.9900	23.2115
			SA	1	0.0004	0	2.1389	1.5295
			QA-100k	1	0.0002	0	1.7493	8.6015
			QA-1M	1	0.0002	0	1.6748	58.2885
sento2	30	60	HPSOGO	1	0.0050	11	4.8425	23.3240
			SA	1	0.0015	1	2.3916	1.5775
			QA-100k	1	0.0010	0	2.1633	7.2080
			QA-1M	1	0.0003	0	1.7320	53.9605
weing1	2	28	HPSOGO	1	0.0000	0	0.0000	1.9630
			SA	1	0.0000	0	0.0000	0.8115
			QA-100k	1	0.0064	0	30.9028	4.0190
			QA-1M	1	0.0033	0	28.0629	34.7420
weing2	2	28	HPSOGO	1	0.0000	0	0.0000	1.9515
			SA	1	0.0000	0	0.0000	0.9120
			QA-100k	1	0.0253	0	52.7646	3.8715
			QA-1M	1	0.0268	0	53.0564	32.3570

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
weing3	2	28	HPSOGO	1	0.0041	0	19.2873	1.9555
			SA	1	0.0000	0	0.0000	0.7860
			QA-100k	1	0.0085	0	21.2087	3.8050
			QA-1M	1	0.0070	0	18.3269	29.3455
weing4	2	28	HPSOGO	1	0.0044	0	29.9003	1.9710
			SA	1	0.0000	0	0.0000	0.8525
			QA-100k	1	0.0092	0	34.7415	3.8525
			QA-1M	1	0.0013	0	16.5545	29.3850
weing5	2	28	HPSOGO	1	0.0082	0	35.3929	1.9735
			SA	1	0.0000	0	0.0000	0.9235
			QA-100k	1	0.0343	0	23.8523	3.7310
			QA-1M	1	0.0363	3009	16.9041	28.6445
weing6	2	28	HPSOGO	1	0.0015	0	13.9642	1.9435
			SA	1	0.0000	0	0.0000	0.9055
			QA-100k	1	0.0383	890	43.3122	3.8840
			QA-1M	1	0.0304	0	48.0137	29.0825
weing7	2	105	HPSOGO	1	0.0078	4354	51.2013	7.1530
			SA	1	0.0009	239	13.2961	2.1270
			QA-100k	1	0.0038	1885	33.2683	9.0485
			QA-1M	1	0.0029	410	34.3737	62.5570
weing8	2	105	HPSOGO	1	0.0602	16424	99.7067	7.6575
			SA	1	0.0198	0	116.2410	1.4200
			QA-100k	1	0.0128	3447	69.5781	5.6240
			QA-1M	1	0.0080	4259	35.3161	39.4870
weish01	5	30	HPSOGO	1	0.0025	0	4.4311	3.6920
			SA	1	0.0000	0	0.0000	0.9825
			QA-100k	1	0.0016	0	3.6497	4.0040
			QA-1M	1	0.0008	0	2.6514	30.0685
weish02	5	30	HPSOGO	1	0.0001	0	0.6892	3.7965
			SA	1	0.0000	0	0.0000	0.7375
			QA-100k	1	0.0000	0	0.0000	3.9740
			QA-1M	1	0.0000	0	0.0000	29.9025
weish03	5	30	HPSOGO	1	0.0044	0	5.0408	3.7535
			SA	1	0.0000	0	0.0000	0.8525
			QA-100k	1	0.0004	0	1.8235	4.0685
			QA-1M	1	0.0006	0	2.1354	30.1845
weish04	5	30	HPSOGO	1	0.0000	0	0.0000	3.8065
			SA	1	0.0000	0	0.0000	1.0885
			QA-100k	1	0.0000	0	0.0000	3.8025
			QA-1M	1	0.0000	0	0.0000	29.2170
weish05	5	30	HPSOGO	1	0.0000	0	0.0000	3.6380
			SA	1	0.0000	0	0.0000	1.0510
			QA-100k	1	0.0000	0	0.0000	3.9905
			QA-1M	1	0.0000	0	0.0000	29.3920

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
weish06	5	40	HPSOGO	1	0.0021	0	3.7310	5.1215
			SA	1	0.0001	0	1.1113	0.9105
			QA-100k	1	0.0000	0	0.0000	4.3475
			QA-1M	1	0.0000	0	0.0000	31.0575
weish07	5	40	HPSOGO	1	0.0015	0	3.2125	5.1605
			SA	1	0.0000	0	0.0000	1.0785
			QA-100k	1	0.0000	0	0.0000	4.2505
			QA-1M	1	0.0000	0	0.0000	29.9295
weish08	5	40	HPSOGO	1	0.0013	0	3.2016	5.5495
			SA	1	0.0000	0	0.0000	1.0435
			QA-100k	1	0.0005	0	2.0199	4.1560
			QA-1M	1	0.0003	0	1.5427	28.9450
weish09	5	40	HPSOGO	1	0.0006	0	2.4739	5.6345
			SA	1	0.0000	0	0.0000	1.0475
			QA-100k	1	0.0022	0	4.5891	4.3120
			QA-1M	1	0.0006	0	2.4850	30.7035
weish10	5	50	HPSOGO	1	0.0047	0	5.0813	7.0170
			SA	1	0.0004	0	2.2450	1.0110
			QA-100k	1	0.0000	0	0.0000	4.8865
			QA-1M	1	0.0000	0	0.0000	34.6890
weish11	5	50	HPSOGO	1	0.0054	0	5.7567	6.8530
			SA	1	0.0000	0	0.0000	1.0900
			QA-100k	1	0.0000	0	0.0000	4.7560
			QA-1M	1	0.0000	0	0.0000	34.0335
weish12	5	50	HPSOGO	1	0.0037	0	5.1010	6.9370
			SA	1	0.0002	0	1.4071	1.0410
			QA-100k	1	0.0000	0	0.0000	4.8020
			QA-1M	1	0.0000	0	0.0000	35.4420
weish13	5	50	HPSOGO	1	0.0033	0	5.3315	7.1970
			SA	1	0.0000	0	0.0000	1.0340
			QA-100k	1	0.0000	0	0.0000	4.8850
			QA-1M	1	0.0000	0	0.0000	34.5695
weish14	5	60	HPSOGO	1	0.0030	0	4.9815	8.4890
			SA	1	0.0011	0	3.4533	1.1750
			QA-100k	1	0.0000	0	0.0000	4.9820
			QA-1M	1	0.0000	0	0.0000	35.1740
weish15	5	60	HPSOGO	1	0.0042	0	4.4164	8.8110
			SA	1	0.0002	0	1.6309	1.2645
			QA-100k	1	0.0000	0	0.0000	4.7585
			QA-1M	1	0.0000	0	0.0000	33.2020
weish16	5	60	HPSOGO	1	0.0036	0	4.8985	8.4575
			SA	1	0.0016	0	3.9497	1.1405
			QA-100k	1	0.0013	0	3.3347	4.7870
			QA-1M	1	0.0035	0	5.4120	33.4235

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
weish17	5	60	HPSOGO	1	0.0014	0	2.8609	7.3915
			SA	1	0.0004	0	2.1564	1.2020
			QA-100k	1	0.0000	0	0.0000	4.8125
			QA-1M	1	0.0000	0	0.0000	33.3565
weish18	5	70	HPSOGO	1	0.0055	15	4.6691	8.6170
			SA	1	0.0011	0	3.5461	1.2075
			QA-100k	1	0.0000	0	0.0000	5.0740
			QA-1M	1	0.0000	0	0.0000	35.7920
weish19	5	70	HPSOGO	1	0.0138	0	7.6118	7.5140
			SA	1	0.0020	0	4.1689	1.2280
			QA-100k	1	0.0010	0	2.4980	5.3040
			QA-1M	1	0.0014	0	3.0594	38.0690
weish20	5	70	HPSOGO	1	0.0067	0	6.4884	8.5820
			SA	1	0.0009	0	3.1631	1.1700
			QA-100k	1	0.0006	0	2.0833	5.1220
			QA-1M	1	0.0004	0	2.0248	35.9335
weish21	5	70	HPSOGO	1	0.0091	0	6.5154	9.2385
			SA	1	0.0020	0	4.7969	1.2745
			QA-100k	1	0.0038	0	3.7947	5.1855
			QA-1M	1	0.0042	0	3.7209	37.0505
weish22	5	80	HPSOGO	1	0.0162	0	8.4496	9.5940
			SA	1	0.0063	0	5.5633	1.4115
			QA-100k	1	0.0067	50	2.6125	5.5915
			QA-1M	1	0.0065	0	3.1305	39.6050
weish23	5	80	HPSOGO	1	0.0218	3	8.6602	10.2490
			SA	1	0.0053	0	6.2849	1.3695
			QA-100k	1	0.0043	3	6.5399	5.7185
			QA-1M	1	0.0003	0	0.7348	42.3705
weish24	5	80	HPSOGO	1	0.0090	5	6.4187	10.6455
			SA	1	0.0024	0	4.6947	1.4275
			QA-100k	1	0.0003	0	1.5083	5.5520
			QA-1M	1	0.0001	0	1.3693	38.6340
weish25	5	80	HPSOGO	1	0.0083	0	5.7053	10.7200
			SA	1	0.0024	0	4.6109	1.2835
			QA-100k	1	0.0016	0	2.3875	5.6360
			QA-1M	1	0.0018	0	3.7523	40.0465
weish26	5	90	HPSOGO	1	0.0227	78	8.1954	11.5615
			SA	1	0.0041	6	3.7550	1.3870
			QA-100k	1	0.0034	32	1.1113	5.8530
			QA-1M	1	0.0035	32	1.8207	41.9150
weish27	5	90	HPSOGO	1	0.0283	121	8.1762	11.2750
			SA	1	0.0064	0	7.5465	1.3745
			QA-100k	1	0.0116	91	4.6087	5.9340
			QA-1M	1	0.0095	0	4.2226	42.7010

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
weish28	5	90	HPSOGO	1	0.0271	77	9.5499	12.3310
			SA	1	0.0042	0	5.6476	1.4140
			QA-100k	1	0.0106	0	7.1025	5.4480
			QA-1M	1	0.0065	0	7.6381	41.3295
weish29	5	90	HPSOGO	1	0.0278	78	8.1548	11.9025
			SA	1	0.0078	0	7.5366	1.3430
			QA-100k	1	0.0117	56	5.5714	5.9320
			QA-1M	1	0.0075	0	5.1788	41.4815
weish30	5	90	HPSOGO	1	0.0106	26	6.5161	11.9075
			SA	1	0.0032	0	3.5036	1.3855
			QA-100k	1	0.0023	0	3.1464	6.0470
			QA-1M	1	0.0026	0	3.0397	42.2250

A.2 Chu-Beasley benchmark

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR10x100-0.25_10	10	100	HPSOGO	1	0.0920	1626	18.2357	20.9270
			SA	1	0.0321	411	11.6297	1.2950
			RQA-100	1	0.0179	274	7.8835	56.6995
OR10x100-0.25_1	10	100	HPSOGO	1	0.0859	1375	15.0173	20.0295
			SA	1	0.0278	387	11.1580	1.3000
			RQA-100	1	0.0068	9	10.6410	57.5160
OR10x100-0.25_2	10	100	HPSOGO	1	0.0879	1265	16.5179	22.7385
			SA	1	0.0291	401	11.5961	1.3010
			RQA-100	1	0.0138	232	7.9869	58.2665
OR10x100-0.25_3	10	100	HPSOGO	1	0.0917	1535	16.4058	20.4985
			SA	1	0.0277	265	12.1188	1.3660
			RQA-100	1	0.0087	66	9.1367	56.7655
OR10x100-0.25_4	10	100	HPSOGO	1	0.0848	1300	17.4637	19.4920
			SA	1	0.0301	412	10.8208	1.3075
			RQA-100	1	0.0136	171	7.8708	57.9960
OR10x100-0.25_5	10	100	HPSOGO	1	0.0950	1533	15.7664	20.7145
			SA	1	0.0288	124	11.6473	1.3000
			RQA-100	1	0.0115	124	9.5729	58.1900
OR10x100-0.25_6	10	100	HPSOGO	1	0.0826	1502	15.5119	19.5335
			SA	1	0.0318	511	11.1043	1.3300
			RQA-100	1	0.0141	147	8.2158	56.5400
OR10x100-0.25_7	10	100	HPSOGO	1	0.0961	1402	18.1718	18.4175
			SA	1	0.0248	178	11.9436	1.2970
			RQA-100	1	0.0102	90	6.8702	55.3970

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR10x100-0.25_8	10	100	HPSOGO	1	0.0964	1690	16.2831	18.4270
			SA	1	0.0278	381	10.7944	1.3040
			RQA-100	1	0.0118	118	8.7086	57.0590
OR10x100-0.25_9	10	100	HPSOGO	1	0.0948	1249	20.2040	17.9445
			SA	1	0.0265	294	12.2209	1.2940
			RQA-100	1	0.0104	122	5.1986	55.9435
OR10x100-0.50_10	10	100	HPSOGO	1	0.0483	1174	17.3868	18.0340
			SA	1	0.0186	574	9.4942	1.3110
			RQA-100	1	0.0036	73	6.5345	63.2835
OR10x100-0.50_1	10	100	HPSOGO	1	0.0484	1579	13.5801	17.6600
			SA	1	0.0187	325	15.1732	1.3090
			RQA-100	1	0.0026	7	4.1623	63.0395
OR10x100-0.50_2	10	100	HPSOGO	1	0.0468	1432	15.8327	18.7200
			SA	1	0.0125	239	11.6447	1.3190
			RQA-100	1	0.0048	143	5.2378	63.3390
OR10x100-0.50_3	10	100	HPSOGO	1	0.0490	1578	14.8950	16.6820
			SA	1	0.0135	154	12.8413	1.3105
			RQA-100	1	0.0029	54	6.9857	66.4715
OR10x100-0.50_4	10	100	HPSOGO	1	0.0430	1341	16.5514	16.4995
			SA	1	0.0185	186	13.8290	1.3110
			RQA-100	1	0.0118	475	4.8631	67.3220
OR10x100-0.50_5	10	100	HPSOGO	1	0.0473	1330	16.3756	16.7405
			SA	1	0.0163	280	11.5217	1.3115
			RQA-100	1	0.0031	0	5.6903	63.5320
OR10x100-0.50_6	10	100	HPSOGO	1	0.0454	1172	18.2277	16.0160
			SA	1	0.0155	296	12.7256	1.3080
			RQA-100	1	0.0039	109	6.6483	63.8850
OR10x100-0.50_7	10	100	HPSOGO	1	0.0459	1155	18.2466	15.8395
			SA	1	0.0171	357	14.3753	1.3180
			RQA-100	1	0.0048	119	6.1604	64.4070
OR10x100-0.50_8	10	100	HPSOGO	1	0.0462	743	17.5647	15.9050
			SA	1	0.0136	166	14.2741	1.3240
			RQA-100	1	0.0044	96	7.3280	66.1570
OR10x100-0.50_9	10	100	HPSOGO	1	0.0513	1672	16.1972	16.1290
			SA	1	0.0140	273	12.9482	1.3155
			RQA-100	1	0.0027	52	6.9444	64.1375
OR10x100-0.75_10	10	100	HPSOGO	1	0.0269	1023	17.6832	16.9780
			SA	1	0.0056	122	9.8742	1.4740
			RQA-100	1	0.0001	0	2.4331	66.4410
OR10x100-0.75_1	10	100	HPSOGO	1	0.0277	916	16.5788	18.6885
			SA	1	0.0075	0	11.3128	1.3040
			RQA-100	1	0.0019	0	6.8586	69.5350

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR10x100-0.75_2	10	100	HPSOGO	1	0.0258	1095	15.5242	20.7400
			SA	1	0.0073	124	10.7515	1.3140
			RQA-100	1	0.0009	0	7.1764	64.2580
OR10x100-0.75_3	10	100	HPSOGO	1	0.0256	1005	15.7797	20.6295
			SA	1	0.0083	80	11.5070	1.3630
			RQA-100	1	0.0015	46	4.4844	62.9135
OR10x100-0.75_4	10	100	HPSOGO	1	0.0258	955	16.3625	18.3885
			SA	1	0.0082	320	11.9048	1.3465
			RQA-100	1	0.0020	76	4.6723	64.2930
OR10x100-0.75_5	10	100	HPSOGO	1	0.0273	989	16.7301	18.5125
			SA	1	0.0073	6	13.0579	1.3110
			RQA-100	1	0.0000	0	1.5232	67.3280
OR10x100-0.75_6	10	100	HPSOGO	1	0.0275	1160	16.2450	18.1940
			SA	1	0.0093	304	11.4782	1.3050
			RQA-100	1	0.0021	83	4.8317	63.0790
OR10x100-0.75_7	10	100	HPSOGO	1	0.0269	1158	14.9377	18.5065
			SA	1	0.0081	204	9.2068	1.3015
			RQA-100	1	0.0016	26	7.3130	63.1600
OR10x100-0.75_8	10	100	HPSOGO	1	0.0279	878	16.8315	17.5010
			SA	1	0.0078	59	12.2229	1.3065
			RQA-100	1	0.0011	0	6.8600	64.0675
OR10x100-0.75_9	10	100	HPSOGO	1	0.0269	984	17.3234	18.3760
			SA	1	0.0075	186	11.0984	1.3085
			RQA-100	1	0.0022	0	6.0655	66.0285
OR10x250-0.25_10	10	250	HPSOGO	1	0.1694	7634	29.0563	48.1235
			SA	1	0.0393	1957	16.1623	2.1450
			RQA-100	1	0.0120	514	9.2715	104.6290
OR10x250-0.25_1	10	250	HPSOGO	1	0.1664	7983	27.9589	45.7035
			SA	1	0.0326	1497	15.4447	2.1490
			RQA-100	1	0.0120	516	7.6896	108.6720
OR10x250-0.25_2	10	250	HPSOGO	1	0.1637	8576	26.1441	44.8245
			SA	1	0.0394	1736	17.0499	2.1560
			RQA-100	1	0.0104	473	7.4646	104.8760
OR10x250-0.25_3	10	250	HPSOGO	1	0.1681	8251	25.3104	42.8510
			SA	1	0.0389	1360	19.8969	2.1445
			RQA-100	1	0.0093	310	7.9057	107.4570
OR10x250-0.25_4	10	250	HPSOGO	1	0.1635	8716	25.5774	39.7680
			SA	1	0.0358	1438	17.8132	2.2010
			RQA-100	1	0.0122	635	8.0281	101.6680
OR10x250-0.25_5	10	250	HPSOGO	1	0.1641	8304	27.2863	40.8235
			SA	1	0.0319	1225	16.5153	2.2380
			RQA-100	1	0.0104	491	7.3451	106.4780

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR10x250-0.25_6	10	250	HPSOGO	1	0.1652	8278	25.5783	43.6980
			SA	1	0.0362	1422	16.7952	2.1560
			RQA-100	1	0.0130	587	8.2492	103.0460
OR10x250-0.25_7	10	250	HPSOGO	1	0.1621	7918	26.0526	46.1880
			SA	1	0.0385	1535	21.0787	2.1555
			RQA-100	1	0.0143	750	6.9448	105.1330
OR10x250-0.25_8	10	250	HPSOGO	1	0.1620	8463	23.9854	41.9530
			SA	1	0.0330	1377	16.0885	2.1545
			RQA-100	1	0.0140	548	9.3005	102.9650
OR10x250-0.25_9	10	250	HPSOGO	1	0.1729	8566	25.9887	39.8805
			SA	1	0.0327	1563	14.9385	2.1600
			RQA-100	1	0.0099	418	8.6058	104.0640
OR10x250-0.50_10	10	250	HPSOGO	1	0.0842	7940	23.9977	38.4430
			SA	1	0.0204	1275	20.4142	2.6820
			RQA-100	1	0.0032	251	7.1554	121.8620
OR10x250-0.50_1	10	250	HPSOGO	1	0.0785	7302	21.9509	38.6520
			SA	1	0.0206	1513	21.8142	2.6930
			RQA-100	1	0.0035	214	7.6508	124.3200
OR10x250-0.50_2	10	250	HPSOGO	1	0.0836	7175	26.1550	38.3155
			SA	1	0.0210	1536	19.0899	2.7880
			RQA-100	1	0.0031	171	7.6498	125.6240
OR10x250-0.50_3	10	250	HPSOGO	1	0.0815	7678	26.5688	38.2700
			SA	1	0.0222	1411	18.6703	2.6730
			RQA-100	1	0.0030	236	7.2650	127.3110
OR10x250-0.50_4	10	250	HPSOGO	1	0.0805	7736	20.9561	38.3495
			SA	1	0.0195	1409	18.8136	2.6780
			RQA-100	1	0.0052	450	7.0534	126.1710
OR10x250-0.50_5	10	250	HPSOGO	1	0.0812	7450	21.6421	38.3565
			SA	1	0.0219	1668	20.9933	2.6660
			RQA-100	1	0.0032	172	6.8374	126.8980
OR10x250-0.50_6	10	250	HPSOGO	1	0.0814	7870	24.4369	38.2365
			SA	1	0.0228	1696	18.4644	2.6915
			RQA-100	1	0.0036	166	7.1708	123.8670
OR10x250-0.50_7	10	250	HPSOGO	1	0.0827	7656	21.3249	38.1685
			SA	1	0.0224	1517	24.2348	2.6715
			RQA-100	1	0.0029	218	7.0604	126.8540
OR10x250-0.50_8	10	250	HPSOGO	1	0.0833	7884	19.5029	38.4465
			SA	1	0.0198	1334	21.7509	2.6725
			RQA-100	1	0.0028	186	7.7750	124.0190
OR10x250-0.50_9	10	250	HPSOGO	1	0.0826	7683	23.0747	38.0385
			SA	1	0.0216	1277	20.9764	2.6910
			RQA-100	1	0.0025	158	7.1739	121.3300

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR10x250-0.75_10	10	250	HPSOGO	1	0.0540	7154	22.8364	38.3215
			SA	1	0.0112	1116	15.9355	2.4690
			RQA-100	1	0.0013	138	5.5245	117.0670
OR10x250-0.75_1	10	250	HPSOGO	1	0.0543	6982	23.5497	38.3535
			SA	1	0.0118	1092	18.7102	2.4555
			RQA-100	1	0.0011	55	5.5839	116.2810
OR10x250-0.75_2	10	250	HPSOGO	1	0.0540	7122	24.8465	38.0640
			SA	1	0.0107	810	15.4428	2.4675
			RQA-100	1	0.0011	20	6.8000	114.8530
OR10x250-0.75_3	10	250	HPSOGO	1	0.0537	6975	22.7497	38.4970
			SA	1	0.0103	1087	14.1651	2.4480
			RQA-100	1	0.0012	43	6.5177	118.7660
OR10x250-0.75_4	10	250	HPSOGO	1	0.0519	6496	22.5579	38.4050
			SA	1	0.0100	1127	13.5542	2.4480
			RQA-100	1	0.0015	147	4.8410	115.5160
OR10x250-0.75_5	10	250	HPSOGO	1	0.0516	6671	21.8698	38.0415
			SA	1	0.0106	912	18.7318	2.5920
			RQA-100	1	0.0017	135	5.8673	113.4800
OR10x250-0.75_6	10	250	HPSOGO	1	0.0529	6924	21.0938	38.4960
			SA	1	0.0108	1041	16.5849	2.5860
			RQA-100	1	0.0011	29	7.3512	117.5580
OR10x250-0.75_7	10	250	HPSOGO	1	0.0555	7006	23.2330	38.6165
			SA	1	0.0104	1178	15.5300	2.4595
			RQA-100	1	0.0011	101	4.9371	117.8310
OR10x250-0.75_8	10	250	HPSOGO	1	0.0532	7223	21.5058	38.0865
			SA	1	0.0105	989	17.1267	2.8740
			RQA-100	1	0.0014	117	5.5077	117.8700
OR10x250-0.75_9	10	250	HPSOGO	1	0.0536	6941	20.3899	38.0715
			SA	1	0.0115	1230	16.4451	2.4445
			RQA-100	1	0.0011	105	6.0000	116.9590
OR10x500-0.25_10	10	500	HPSOGO	1	0.2051	22120	29.9164	93.6210
			SA	1	0.0426	4094	22.1522	3.6445
			RQA-100	1	0.0123	1175	10.1440	164.0750
OR10x500-0.25_1	10	500	HPSOGO	1	0.2016	21435	28.3266	88.7420
			SA	1	0.0449	4064	24.5161	3.7565
			RQA-100	1	0.0131	1334	8.3358	177.1430
OR10x500-0.25_2	10	500	HPSOGO	1	0.2029	21995	29.5660	84.3750
			SA	1	0.0421	4215	19.9625	3.7085
			RQA-100	1	0.0109	1131	8.4342	177.9740
OR10x500-0.25_3	10	500	HPSOGO	1	0.2047	20982	32.1692	98.7380
			SA	1	0.0408	3584	25.1707	3.6510
			RQA-100	1	0.0119	1093	10.2896	184.6420

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR10x500-0.25_4	10	500	HPSOGO	1	0.2001	21494	32.0949	106.4380
			SA	1	0.0446	3906	23.5266	3.6705
			RQA-100	1	0.0144	1479	10.0648	182.5020
OR10x500-0.25_5	10	500	HPSOGO	1	0.2013	21228	32.5434	106.9240
			SA	1	0.0420	3470	26.7880	3.6380
			RQA-100	1	0.0114	1153	8.5144	180.9070
OR10x500-0.25_6	10	500	HPSOGO	1	0.2150	23429	29.7296	106.1430
			SA	1	0.0420	3550	20.5188	3.6390
			RQA-100	1	0.0114	1161	9.1848	176.7600
OR10x500-0.25_7	10	500	HPSOGO	1	0.2066	22164	76.4228	94.1900
			SA	1	0.0419	4124	21.2368	3.6455
			RQA-100	1	0.0130	1321	10.5546	179.3070
OR10x500-0.25_8	10	500	HPSOGO	1	0.2034	22482	28.5598	102.3200
			SA	1	0.0396	3537	21.4881	3.6600
			RQA-100	1	0.0116	1085	9.8005	178.9930
OR10x500-0.25_9	10	500	HPSOGO	1	0.1963	19934	29.4571	107.9440
			SA	1	0.0464	3871	24.1887	3.6540
			RQA-100	1	0.0144	1497	10.4857	162.2620
OR10x500-0.50_10	10	500	HPSOGO	1	0.0988	19191	27.3361	107.2180
			SA	1	0.0274	4810	28.3554	5.0305
			RQA-100	1	0.0045	813	8.1817	235.4560
OR10x500-0.50_1	10	500	HPSOGO	1	0.0981	19681	23.4734	109.6880
			SA	1	0.0254	4490	23.4041	4.7650
			RQA-100	1	0.0043	751	8.7564	225.0060
OR10x500-0.50_2	10	500	HPSOGO	1	0.1006	20179	29.6162	109.7070
			SA	1	0.0267	4417	25.6628	4.7920
			RQA-100	1	0.0037	492	9.2817	203.3310
OR10x500-0.50_3	10	500	HPSOGO	1	0.1032	20264	30.8521	106.8740
			SA	1	0.0263	4796	24.9780	4.7660
			RQA-100	1	0.0040	747	8.1477	232.4040
OR10x500-0.50_4	10	500	HPSOGO	1	0.1028	20389	27.2324	107.0760
			SA	1	0.0265	4270	26.6510	4.9810
			RQA-100	1	0.0040	753	7.0640	223.8580
OR10x500-0.50_5	10	500	HPSOGO	1	0.0957	18431	29.1307	100.5890
			SA	1	0.0252	3325	27.4691	4.7585
			RQA-100	1	0.0038	553	8.9666	231.5800
OR10x500-0.50_6	10	500	HPSOGO	1	0.1055	20954	28.8353	104.1000
			SA	1	0.0275	4733	23.7653	4.7965
			RQA-100	1	0.0042	790	7.3519	214.4060
OR10x500-0.50_7	10	500	HPSOGO	1	0.1007	20120	30.6235	95.9880
			SA	1	0.0254	4089	24.4141	4.7660
			RQA-100	1	0.0040	747	8.2934	245.3570

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR10x500-0.50_8	10	500	HPSOGO	1	0.1013	19796	26.9090	94.6400
			SA	1	0.0270	4214	25.1951	4.7470
			RQA-100	1	0.0037	680	7.9473	231.4500
OR10x500-0.50_9	10	500	HPSOGO	1	0.1010	19369	26.9844	93.7460
			SA	1	0.0269	4366	26.4773	4.7770
			RQA-100	1	0.0038	670	8.1854	232.2860
OR10x500-0.75_10	10	500	HPSOGO	1	0.0731	20799	30.4837	94.8360
			SA	1	0.0139	3065	20.5266	4.2015
			RQA-100	1	0.0015	378	6.0332	206.2680
OR10x500-0.75_1	10	500	HPSOGO	1	0.0747	20151	30.7441	93.4220
			SA	1	0.0139	2906	26.2637	4.1895
			RQA-100	1	0.0016	360	6.6843	207.2460
OR10x500-0.75_2	10	500	HPSOGO	1	0.0729	18767	33.4622	94.0215
			SA	1	0.0131	2750	22.2450	4.2435
			RQA-100	1	0.0015	324	6.9671	215.2560
OR10x500-0.75_3	10	500	HPSOGO	1	0.0691	19921	24.3084	88.2710
			SA	1	0.0127	3119	21.3061	4.3560
			RQA-100	1	0.0017	462	4.6363	209.5990
OR10x500-0.75_4	10	500	HPSOGO	1	0.0736	20220	32.8064	89.1480
			SA	1	0.0144	3270	18.0707	4.1785
			RQA-100	1	0.0015	335	6.6170	207.4950
OR10x500-0.75_5	10	500	HPSOGO	1	0.0749	20590	27.6405	85.5165
			SA	1	0.0138	2983	23.2882	4.2135
			RQA-100	1	0.0014	266	7.1764	204.3540
OR10x500-0.75_6	10	500	HPSOGO	1	0.0685	19206	27.8673	86.3160
			SA	1	0.0147	2796	22.0873	4.2210
			RQA-100	1	0.0019	445	7.1659	209.4210
OR10x500-0.75_7	10	500	HPSOGO	1	0.0720	18612	33.7680	90.0835
			SA	1	0.0136	2658	22.0998	4.2500
			RQA-100	1	0.0016	390	6.1984	202.4440
OR10x500-0.75_8	10	500	HPSOGO	1	0.0710	18983	32.0585	92.5415
			SA	1	0.0136	2736	22.6528	4.2320
			RQA-100	1	0.0016	321	7.4044	212.0270
OR10x500-0.75_9	10	500	HPSOGO	1	0.0734	20189	28.8870	94.8675
			SA	1	0.0119	2537	22.6128	4.1945
			RQA-100	1	0.0015	323	7.1648	214.3100
OR30x100-0.25_10	30	100	HPSOGO	1	0.0982	1588	17.0287	47.2120
			SA	1	0.0236	78	10.3223	1.6890
			RQA-100	1	0.0156	220	4.2866	73.4645
OR30x100-0.25_1	30	100	HPSOGO	1	0.0960	1002	19.9104	47.5455
			SA	1	0.0250	299	11.4612	1.6515
			RQA-100	1	0.0123	73	6.7639	74.2135

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR30x100-0.25_2	30	100	HPSOGO	1	0.1032	1521	16.5015	44.6165
			SA	1	0.0305	355	12.7059	1.6560
			RQA-100	1	0.0164	288	4.3162	67.4505
OR30x100-0.25_3	30	100	HPSOGO	1	0.0970	1561	15.6775	42.6000
			SA	1	0.0306	387	10.9704	1.6485
			RQA-100	1	0.0193	264	8.3388	69.4355
OR30x100-0.25_4	30	100	HPSOGO	1	0.0923	1248	16.6253	43.5260
			SA	1	0.0253	306	9.4700	1.6660
			RQA-100	1	0.0143	270	6.1563	69.8215
OR30x100-0.25_5	30	100	HPSOGO	1	0.1011	1519	19.4008	45.0090
			SA	1	0.0252	263	11.3206	1.6590
			RQA-100	1	0.0168	342	5.0190	73.2200
OR30x100-0.25_6	30	100	HPSOGO	1	0.0995	1513	16.2069	46.8435
			SA	1	0.0244	74	14.2206	1.6590
			RQA-100	1	0.0092	0	11.4512	68.9355
OR30x100-0.25_7	30	100	HPSOGO	1	0.0952	1693	13.5359	53.8800
			SA	1	0.0297	352	10.9659	1.6510
			RQA-100	1	0.0171	223	6.7617	68.3290
OR30x100-0.25_8	30	100	HPSOGO	1	0.1033	1528	16.4346	53.6630
			SA	1	0.0258	248	12.4900	1.6675
			RQA-100	1	0.0132	172	7.6658	72.3650
OR30x100-0.25_9	30	100	HPSOGO	1	0.0956	1076	17.0294	52.9760
			SA	1	0.0292	334	11.5737	1.6650
			RQA-100	1	0.0143	248	6.3182	74.0565
OR30x100-0.50_10	30	100	HPSOGO	1	0.0501	1454	17.2841	53.1765
			SA	1	0.0126	347	10.0871	1.6580
			RQA-100	1	0.0042	38	3.6966	75.0115
OR30x100-0.50_1	30	100	HPSOGO	1	0.0525	1576	14.7570	53.2505
			SA	1	0.0175	304	11.2650	1.6550
			RQA-100	1	0.0071	137	6.9878	74.3125
OR30x100-0.50_2	30	100	HPSOGO	1	0.0507	1434	16.1298	53.5080
			SA	1	0.0151	234	14.4972	2.0380
			RQA-100	1	0.0041	4	7.9215	73.0135
OR30x100-0.50_3	30	100	HPSOGO	1	0.0517	1513	17.4770	52.1880
			SA	1	0.0169	362	13.1928	2.0250
			RQA-100	1	0.0079	173	9.4101	71.6705
OR30x100-0.50_4	30	100	HPSOGO	1	0.0487	1559	15.5440	49.9520
			SA	1	0.0128	316	11.7924	1.7655
			RQA-100	1	0.0025	69	7.2990	74.3485
OR30x100-0.50_5	30	100	HPSOGO	1	0.0518	1338	17.5704	47.1200
			SA	1	0.0175	440	13.2469	1.6700
			RQA-100	1	0.0040	44	8.1437	70.8510

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR30x100-0.50_6	30	100	HPSOGO	1	0.0475	1031	15.3251	49.2690
			SA	1	0.0156	297	12.2963	1.9760
			RQA-100	1	0.0019	0	6.7112	69.2775
OR30x100-0.50_7	30	100	HPSOGO	1	0.0520	1465	17.3534	53.6570
			SA	1	0.0160	310	13.6220	2.0380
			RQA-100	1	0.0044	81	5.2062	72.0365
OR30x100-0.50_8	30	100	HPSOGO	1	0.0521	1717	16.6942	52.4905
			SA	1	0.0162	219	15.7987	1.6675
			RQA-100	1	0.0025	50	5.6258	69.6505
OR30x100-0.50_9	30	100	HPSOGO	1	0.0522	1620	15.3281	54.1725
			SA	1	0.0183	415	15.1493	1.6650
			RQA-100	1	0.0057	239	1.5149	73.9725
OR30x100-0.75_10	30	100	HPSOGO	1	0.0314	1434	14.0926	53.9540
			SA	1	0.0083	180	11.4804	1.6350
			RQA-100	1	0.0011	0	6.8949	72.4805
OR30x100-0.75_1	30	100	HPSOGO	1	0.0314	1479	14.9382	53.9295
			SA	1	0.0072	0	12.2229	1.6360
			RQA-100	1	0.0000	0	0.0000	75.1210
OR30x100-0.75_2	30	100	HPSOGO	1	0.0295	1322	14.0650	53.7985
			SA	1	0.0075	214	10.5880	1.6395
			RQA-100	1	0.0014	0	7.7330	72.1940
OR30x100-0.75_3	30	100	HPSOGO	1	0.0292	932	16.6190	53.7800
			SA	1	0.0077	74	11.5449	1.8650
			RQA-100	1	0.0022	60	6.6348	72.5210
OR30x100-0.75_4	30	100	HPSOGO	1	0.0282	1039	16.9071	53.6220
			SA	1	0.0073	264	9.9303	1.6550
			RQA-100	1	0.0027	11	5.1884	72.7030
OR30x100-0.75_5	30	100	HPSOGO	1	0.0309	1235	16.3727	52.9735
			SA	1	0.0088	275	12.8649	1.6245
			RQA-100	1	0.0039	93	6.4869	75.7725
OR30x100-0.75_6	30	100	HPSOGO	1	0.0270	1093	15.1494	52.9675
			SA	1	0.0069	97	12.1466	1.6685
			RQA-100	1	0.0031	123	3.8872	72.8995
OR30x100-0.75_7	30	100	HPSOGO	1	0.0291	1300	14.1723	53.8865
			SA	1	0.0069	28	12.2572	1.6305
			RQA-100	1	0.0008	0	6.6464	80.4310
OR30x100-0.75_8	30	100	HPSOGO	1	0.0315	1233	16.3421	53.7230
			SA	1	0.0080	253	10.6677	1.7185
			RQA-100	1	0.0021	0	5.5045	74.6610
OR30x100-0.75_9	30	100	HPSOGO	1	0.0302	1163	13.2123	52.9865
			SA	1	0.0078	155	11.5698	1.6370
			RQA-100	1	0.0026	88	6.1449	70.9815

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR30x250-0.25_10	30	250	HPSOGO	1	0.1754	8405	24.5932	112.9900
			SA	1	0.0320	1223	18.4293	2.5150
			RQA-100	1	0.0112	487	9.1477	106.3000
OR30x250-0.25_1	30	250	HPSOGO	1	0.1846	9781	19.7489	106.0350
			SA	1	0.0357	1263	16.8181	2.4585
			RQA-100	1	0.0134	572	9.9401	108.1050
OR30x250-0.25_2	30	250	HPSOGO	1	0.1810	9144	26.8812	110.9750
			SA	1	0.0385	1873	14.6301	2.4710
			RQA-100	1	0.0194	914	9.4345	106.6910
OR30x250-0.25_3	30	250	HPSOGO	1	0.1874	9836	22.2636	130.4400
			SA	1	0.0334	1475	16.3505	2.4825
			RQA-100	1	0.0091	372	8.7977	105.8830
OR30x250-0.25_4	30	250	HPSOGO	1	0.1838	9173	28.4114	129.8800
			SA	1	0.0363	1567	16.7975	2.5295
			RQA-100	1	0.0110	452	9.6638	104.9300
OR30x250-0.25_5	30	250	HPSOGO	1	0.1749	7595	29.8278	128.8150
			SA	1	0.0364	1520	18.2181	2.4680
			RQA-100	1	0.0128	617	6.2578	105.4630
OR30x250-0.25_6	30	250	HPSOGO	1	0.1796	8411	24.5153	120.9470
			SA	1	0.0329	1187	17.0422	2.4720
			RQA-100	1	0.0114	533	8.4226	106.8710
OR30x250-0.25_7	30	250	HPSOGO	1	0.1737	8662	23.5128	130.9980
			SA	1	0.0389	1315	15.9906	2.4895
			RQA-100	1	0.0131	571	8.6562	108.2430
OR30x250-0.25_8	30	250	HPSOGO	1	0.1764	8708	23.0794	133.4050
			SA	1	0.0364	1264	19.2702	2.4850
			RQA-100	1	0.0093	335	7.5103	104.4860
OR30x250-0.25_9	30	250	HPSOGO	1	0.1659	8129	25.7125	131.1210
			SA	1	0.0393	1423	18.8616	2.4745
			RQA-100	1	0.0114	552	7.8070	105.9220
OR30x250-0.50_10	30	250	HPSOGO	1	0.0864	8083	21.4199	131.9460
			SA	1	0.0200	1125	22.4288	3.0080
			RQA-100	1	0.0046	393	7.2284	126.8770
OR30x250-0.50_1	30	250	HPSOGO	1	0.0898	8354	24.1173	130.1920
			SA	1	0.0202	1530	18.4949	2.9970
			RQA-100	1	0.0044	356	7.5766	128.0230
OR30x250-0.50_2	30	250	HPSOGO	1	0.0885	8345	24.9028	125.8910
			SA	1	0.0200	1198	20.7699	2.9905
			RQA-100	1	0.0036	246	6.7305	127.2860
OR30x250-0.50_3	30	250	HPSOGO	1	0.0840	7601	22.9532	124.1520
			SA	1	0.0214	1534	19.1872	3.3085
			RQA-100	1	0.0036	265	6.6310	125.6190

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR30x250-0.50_4	30	250	HPSOGO	1	0.0875	8262	28.1420	116.9590
			SA	1	0.0207	1332	18.9897	3.0205
			RQA-100	1	0.0043	214	7.8013	127.5250
OR30x250-0.50_5	30	250	HPSOGO	1	0.0868	8271	21.2818	113.2880
			SA	1	0.0193	1357	20.9800	3.0185
			RQA-100	1	0.0037	305	6.8556	129.8750
OR30x250-0.50_6	30	250	HPSOGO	1	0.0846	7710	21.8211	113.8770
			SA	1	0.0209	1405	18.8600	3.0750
			RQA-100	1	0.0050	325	8.7567	126.2370
OR30x250-0.50_7	30	250	HPSOGO	1	0.0877	7791	22.5785	114.7950
			SA	1	0.0226	1371	17.4780	3.0110
			RQA-100	1	0.0046	319	7.3280	126.3120
OR30x250-0.50_8	30	250	HPSOGO	1	0.0893	7662	22.5526	108.7390
			SA	1	0.0203	1458	19.7647	3.0000
			RQA-100	1	0.0032	242	6.6933	131.0910
OR30x250-0.50_9	30	250	HPSOGO	1	0.0865	8247	20.9619	106.4230
			SA	1	0.0227	1120	22.4695	3.0805
			RQA-100	1	0.0038	202	7.4927	129.2660
OR30x250-0.75_10	30	250	HPSOGO	1	0.0554	6891	24.0816	105.6860
			SA	1	0.0115	1110	17.9597	2.7970
			RQA-100	1	0.0021	180	6.7450	118.3380
OR30x250-0.75_1	30	250	HPSOGO	1	0.0598	7354	22.0307	106.1330
			SA	1	0.0109	834	17.3774	2.7965
			RQA-100	1	0.0018	144	6.1725	114.1570
OR30x250-0.75_2	30	250	HPSOGO	1	0.0564	7652	21.1948	108.8440
			SA	1	0.0104	998	18.6279	2.8255
			RQA-100	1	0.0019	131	6.8877	113.0990
OR30x250-0.75_3	30	250	HPSOGO	1	0.0590	7761	23.2855	113.4170
			SA	1	0.0119	750	20.7764	2.8040
			RQA-100	1	0.0014	149	5.5335	111.7330
OR30x250-0.75_4	30	250	HPSOGO	1	0.0572	7617	24.2997	109.2220
			SA	1	0.0119	1125	19.5172	2.8175
			RQA-100	1	0.0020	207	5.6772	112.1840
OR30x250-0.75_5	30	250	HPSOGO	1	0.0551	6754	22.3505	103.6630
			SA	1	0.0102	953	16.3907	2.8425
			RQA-100	1	0.0019	0	6.1919	114.1830
OR30x250-0.75_6	30	250	HPSOGO	1	0.0541	6818	21.3192	104.3540
			SA	1	0.0105	897	17.2583	2.8740
			RQA-100	1	0.0017	166	6.2085	112.3150
OR30x250-0.75_7	30	250	HPSOGO	1	0.0582	7431	24.4683	103.4470
			SA	1	0.0116	1094	20.9143	2.8745
			RQA-100	1	0.0013	65	5.6701	110.2540

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR30x250-0.75_8	30	250	HPSOGO	1	0.0557	7291	25.2086	103.5260
			SA	1	0.0112	911	20.4685	2.8025
			RQA-100	1	0.0017	144	6.3344	114.0020
OR30x250-0.75_9	30	250	HPSOGO	1	0.0544	7499	20.3374	110.3560
			SA	1	0.0111	918	21.5824	2.7965
			RQA-100	1	0.0016	188	6.3119	116.5520
OR30x500-0.25_10	30	500	HPSOGO	1	0.2147	23321	31.7777	219.5610
			SA	1	0.0432	3296	27.1382	3.9370
			RQA-100	1	0.0122	1128	9.9938	179.8810
OR30x500-0.25_1	30	500	HPSOGO	1	0.2097	21551	32.9275	241.4460
			SA	1	0.0411	3601	25.0619	3.9730
			RQA-100	1	0.0135	1232	10.4805	156.9070
OR30x500-0.25_2	30	500	HPSOGO	1	0.2116	21506	30.5462	256.0100
			SA	1	0.0455	4017	30.1471	4.4045
			RQA-100	1	0.0108	927	10.7713	184.2570
OR30x500-0.25_3	30	500	HPSOGO	1	0.2119	22829	25.7963	252.5550
			SA	1	0.0459	3890	23.8789	3.9625
			RQA-100	1	0.0121	1061	10.4833	171.4910
OR30x500-0.25_4	30	500	HPSOGO	1	0.1995	20112	33.9659	257.9420
			SA	1	0.0462	4454	22.5212	3.9745
			RQA-100	1	0.0118	1202	9.4242	170.9690
OR30x500-0.25_5	30	500	HPSOGO	1	0.2018	21295	30.6346	261.0770
			SA	1	0.0419	3675	21.4089	4.1090
			RQA-100	1	0.0118	976	10.9343	181.0010
OR30x500-0.25_6	30	500	HPSOGO	1	0.2077	22789	26.7724	249.2330
			SA	1	0.0449	3749	29.5397	3.9350
			RQA-100	1	0.0157	1304	11.8004	177.5640
OR30x500-0.25_7	30	500	HPSOGO	1	0.2086	21368	28.5850	229.3330
			SA	1	0.0405	3097	24.0734	3.9440
			RQA-100	1	0.0133	1262	9.5134	172.8850
OR30x500-0.25_8	30	500	HPSOGO	1	0.2114	21736	30.0924	226.7200
			SA	1	0.0396	3478	20.4989	3.9450
			RQA-100	1	0.0123	1236	7.5040	187.5480
OR30x500-0.25_9	30	500	HPSOGO	1	0.2045	21448	29.6737	217.5080
			SA	1	0.0414	3951	21.9367	3.9480
			RQA-100	1	0.0139	1311	10.2742	183.9410
OR30x500-0.50_10	30	500	HPSOGO	1	0.1068	21841	24.9921	210.9530
			SA	1	0.0252	3991	26.8254	5.0830
			RQA-100	1	0.0044	607	9.3220	216.5470
OR30x500-0.50_1	30	500	HPSOGO	1	0.1093	22080	25.4244	219.7930
			SA	1	0.0264	3863	30.5663	5.1305
			RQA-100	1	0.0037	664	7.5598	217.8530

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR30x500-0.50_2	30	500	HPSOGO	1	0.1030	20449	27.0585	211.8460
			SA	1	0.0288	4528	28.4035	5.0725
			RQA-100	1	0.0049	845	8.4487	220.2850
OR30x500-0.50_3	30	500	HPSOGO	1	0.1102	21476	25.9480	206.1260
			SA	1	0.0281	4296	31.5963	5.1645
			RQA-100	1	0.0042	750	7.7418	201.8760
OR30x500-0.50_4	30	500	HPSOGO	1	0.1076	21983	24.6576	212.7020
			SA	1	0.0273	3861	31.1565	5.0850
			RQA-100	1	0.0047	845	8.5756	205.0760
OR30x500-0.50_5	30	500	HPSOGO	1	0.1091	21389	28.9430	200.6850
			SA	1	0.0276	3972	32.6846	5.0875
			RQA-100	1	0.0043	789	6.9491	215.2090
OR30x500-0.50_6	30	500	HPSOGO	1	0.1047	21112	28.0393	200.1500
			SA	1	0.0254	4129	27.0967	5.1645
			RQA-100	1	0.0046	832	8.6371	219.2260
OR30x500-0.50_7	30	500	HPSOGO	1	0.1101	22025	24.4786	190.0310
			SA	1	0.0280	4573	23.5243	5.4760
			RQA-100	1	0.0036	625	8.1532	212.6360
OR30x500-0.50_8	30	500	HPSOGO	1	0.1074	21985	26.0108	189.6150
			SA	1	0.0278	3837	30.4765	5.1680
			RQA-100	1	0.0048	859	9.2596	221.8600
OR30x500-0.50_9	30	500	HPSOGO	1	0.1063	20978	30.7942	193.6430
			SA	1	0.0287	4255	29.3241	5.1285
			RQA-100	1	0.0044	606	8.8065	209.5060
OR30x500-0.75_10	30	500	HPSOGO	1	0.0730	19990	34.0404	188.3080
			SA	1	0.0149	3085	23.4921	4.5685
			RQA-100	1	0.0021	498	6.9455	196.7860
OR30x500-0.75_1	30	500	HPSOGO	1	0.0750	18754	28.2254	187.8940
			SA	1	0.0137	3257	23.6025	4.5720
			RQA-100	1	0.0020	440	7.7685	204.1960
OR30x500-0.75_2	30	500	HPSOGO	1	0.0724	20278	28.2024	184.5870
			SA	1	0.0147	3016	27.7110	4.6315
			RQA-100	1	0.0019	464	7.5236	195.9630
OR30x500-0.75_3	30	500	HPSOGO	1	0.0775	20810	34.0164	183.4460
			SA	1	0.0130	2359	27.9778	4.5465
			RQA-100	1	0.0016	346	7.7330	210.5900
OR30x500-0.75_4	30	500	HPSOGO	1	0.0743	20769	31.4002	184.1070
			SA	1	0.0150	3330	26.6785	4.6045
			RQA-100	1	0.0022	480	7.6453	198.7770
OR30x500-0.75_5	30	500	HPSOGO	1	0.0736	19259	32.5138	184.1290
			SA	1	0.0140	2947	24.7285	4.7280
			RQA-100	1	0.0021	513	7.8905	198.9930

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR30x500-0.75_6	30	500	HPSOGO	1	0.0738	19510	35.0366	183.8680
			SA	1	0.0140	3144	25.4335	4.6545
			RQA-100	1	0.0022	532	6.8702	195.4090
OR30x500-0.75_7	30	500	HPSOGO	1	0.0719	19514	30.1385	184.0370
			SA	1	0.0158	3627	24.5474	4.9900
			RQA-100	1	0.0019	451	7.1896	197.0200
OR30x500-0.75_8	30	500	HPSOGO	1	0.0768	22155	26.7292	191.6010
			SA	1	0.0123	2746	21.3776	4.7545
			RQA-100	1	0.0017	337	8.1117	208.3460
OR30x500-0.75_9	30	500	HPSOGO	1	0.0736	20288	34.4812	184.5400
			SA	1	0.0141	2726	23.4725	5.1265
			RQA-100	1	0.0019	416	8.4428	198.4100
OR5x100-0.25_10	5	100	HPSOGO	1	0.0713	939	16.5135	10.4165
			SA	1	0.0217	217	11.5178	1.3255
			RQA-100	1	0.0074	133	5.8975	55.1100
OR5x100-0.25_1	5	100	HPSOGO	1	0.0802	1546	15.5108	10.2720
			SA	1	0.0207	192	11.1241	1.3495
			RQA-100	1	0.0114	128	10.6705	52.7020
OR5x100-0.25_2	5	100	HPSOGO	1	0.0814	1606	15.0906	10.1670
			SA	1	0.0247	159	13.0432	1.3940
			RQA-100	1	0.0077	84	9.0222	52.3100
OR5x100-0.25_3	5	100	HPSOGO	1	0.0792	934	16.0181	10.5790
			SA	1	0.0204	114	11.6634	1.2940
			RQA-100	1	0.0056	28	7.2087	53.2445
OR5x100-0.25_4	5	100	HPSOGO	1	0.0769	1293	17.4436	11.3715
			SA	1	0.0237	370	10.5617	1.2215
			RQA-100	1	0.0073	118	6.8688	50.3120
OR5x100-0.25_5	5	100	HPSOGO	1	0.0753	841	18.1846	11.8885
			SA	1	0.0215	238	12.2646	1.4130
			RQA-100	1	0.0076	73	5.7810	52.7625
OR5x100-0.25_6	5	100	HPSOGO	1	0.0851	1343	18.1011	12.9560
			SA	1	0.0245	227	13.4222	1.4110
			RQA-100	1	0.0124	199	7.6030	49.5145
OR5x100-0.25_7	5	100	HPSOGO	1	0.0782	1322	18.7577	13.3400
			SA	1	0.0255	423	11.4449	1.4320
			RQA-100	1	0.0122	160	9.3408	49.2225
OR5x100-0.25_8	5	100	HPSOGO	1	0.0829	952	18.8280	13.0495
			SA	1	0.0258	386	12.6819	1.1995
			RQA-100	1	0.0061	80	6.6159	51.1165
OR5x100-0.25_9	5	100	HPSOGO	1	0.0748	864	16.7055	13.2460
			SA	1	0.0247	252	12.6933	1.2325
			RQA-100	1	0.0041	0	5.5136	49.5480

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR5x100-0.50_10	5	100	HPSOGO	1	0.0384	1047	15.0781	12.3740
			SA	1	0.0095	170	11.2621	1.2180
			RQA-100	1	0.0017	43	5.9821	57.9785
OR5x100-0.50_1	5	100	HPSOGO	1	0.0415	1262	16.1096	13.9825
			SA	1	0.0148	405	11.8148	1.2010
			RQA-100	1	0.0014	0	6.5391	55.5340
OR5x100-0.50_2	5	100	HPSOGO	1	0.0397	1025	16.4767	14.2385
			SA	1	0.0136	335	12.4467	1.2140
			RQA-100	1	0.0006	0	5.4342	58.1170
OR5x100-0.50_3	5	100	HPSOGO	1	0.0364	1159	14.9845	14.8080
			SA	1	0.0125	243	10.8812	1.2040
			RQA-100	1	0.0010	22	4.8062	54.5915
OR5x100-0.50_4	5	100	HPSOGO	1	0.0401	1247	16.1012	14.1355
			SA	1	0.0106	99	11.5929	1.2075
			RQA-100	1	0.0009	0	5.0695	56.3200
OR5x100-0.50_5	5	100	HPSOGO	1	0.0418	1328	13.5333	12.8190
			SA	1	0.0123	237	12.6317	1.2445
			RQA-100	1	0.0019	26	7.4766	53.3460
OR5x100-0.50_6	5	100	HPSOGO	1	0.0394	1140	16.3707	12.9445
			SA	1	0.0126	253	10.1242	1.2020
			RQA-100	1	0.0024	52	5.1962	57.8395
OR5x100-0.50_7	5	100	HPSOGO	1	0.0402	1258	14.0472	12.3400
			SA	1	0.0147	345	11.4636	1.2070
			RQA-100	1	0.0017	0	6.1677	57.4930
OR5x100-0.50_8	5	100	HPSOGO	1	0.0435	1378	13.4374	11.7480
			SA	1	0.0123	258	12.1947	1.2775
			RQA-100	1	0.0029	77	4.6583	54.4110
OR5x100-0.50_9	5	100	HPSOGO	1	0.0394	1007	16.0800	13.1325
			SA	1	0.0159	414	11.5263	1.3075
			RQA-100	1	0.0032	78	5.4111	55.3675
OR5x100-0.75_10	5	100	HPSOGO	1	0.0254	1182	13.9684	13.7835
			SA	1	0.0074	214	11.2818	1.3010
			RQA-100	1	0.0005	5	5.0498	54.6885
OR5x100-0.75_1	5	100	HPSOGO	1	0.0240	976	14.0950	13.4450
			SA	1	0.0079	167	10.2176	1.2300
			RQA-100	1	0.0005	0	3.9294	56.6895
OR5x100-0.75_2	5	100	HPSOGO	1	0.0232	884	15.8442	11.5450
			SA	1	0.0065	235	8.5598	1.2280
			RQA-100	1	0.0019	0	5.0498	57.0680
OR5x100-0.75_3	5	100	HPSOGO	1	0.0233	905	13.3901	12.0645
			SA	1	0.0053	140	9.0488	1.3685
			RQA-100	1	0.0013	41	5.6877	56.6035

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR5x100-0.75_4	5	100	HPSOGO	1	0.0209	912	13.6066	11.7010
			SA	1	0.0058	145	10.9791	1.2390
			RQA-100	1	0.0007	0	4.3203	57.5250
OR5x100-0.75_5	5	100	HPSOGO	1	0.0255	1042	15.4328	11.8490
			SA	1	0.0064	107	10.7247	1.3595
			RQA-100	1	0.0010	12	3.5777	56.1630
OR5x100-0.75_6	5	100	HPSOGO	1	0.0238	1027	14.9613	11.3965
			SA	1	0.0062	123	9.0407	1.2045
			RQA-100	1	0.0007	0	3.4900	55.6260
OR5x100-0.75_7	5	100	HPSOGO	1	0.0239	768	18.0885	11.5550
			SA	1	0.0065	99	11.1283	1.2125
			RQA-100	1	0.0013	19	5.5911	58.2425
OR5x100-0.75_8	5	100	HPSOGO	1	0.0229	630	17.7206	11.6175
			SA	1	0.0068	144	10.6715	1.2130
			RQA-100	1	0.0002	0	3.9988	56.0585
OR5x100-0.75_9	5	100	HPSOGO	1	0.0254	627	16.6131	11.9035
			SA	1	0.0082	233	10.0539	1.2060
			RQA-100	1	0.0012	0	5.0725	55.9525
OR5x250-0.25_10	5	250	HPSOGO	1	0.1497	7656	22.2030	27.5935
			SA	1	0.0313	1017	17.5699	2.3115
			RQA-100	1	0.0073	345	7.0612	101.9480
OR5x250-0.25_1	5	250	HPSOGO	1	0.1636	8035	23.5741	31.6580
			SA	1	0.0328	1242	16.7637	2.1475
			RQA-100	1	0.0088	432	6.8993	102.7270
OR5x250-0.25_2	5	250	HPSOGO	1	0.1640	8756	27.0703	31.2155
			SA	1	0.0338	1424	18.9182	2.0820
			RQA-100	1	0.0108	501	8.5531	100.2340
OR5x250-0.25_3	5	250	HPSOGO	1	0.1613	8216	24.3247	32.2130
			SA	1	0.0360	1627	18.0294	2.1025
			RQA-100	1	0.0122	561	9.7831	100.4600
OR5x250-0.25_4	5	250	HPSOGO	1	0.1404	7063	25.8493	31.3340
			SA	1	0.0341	1421	16.8083	2.0960
			RQA-100	1	0.0125	602	8.2825	104.2400
OR5x250-0.25_5	5	250	HPSOGO	1	0.1507	7665	23.5893	31.1050
			SA	1	0.0301	1215	15.9720	2.0920
			RQA-100	1	0.0089	372	7.4243	99.2035
OR5x250-0.25_6	5	250	HPSOGO	1	0.1520	7442	27.2694	31.4255
			SA	1	0.0375	1512	17.3094	2.0840
			RQA-100	1	0.0125	561	7.9246	100.3170
OR5x250-0.25_7	5	250	HPSOGO	1	0.1568	7101	27.5392	31.5650
			SA	1	0.0352	1448	19.7321	2.0785
			RQA-100	1	0.0098	322	8.5758	100.9390

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR5x250-0.25_8	5	250	HPSOGO	1	0.1508	6717	27.7852	29.2535
			SA	1	0.0307	1205	17.4671	2.0860
			RQA-100	1	0.0103	503	7.2360	101.4900
OR5x250-0.25_9	5	250	HPSOGO	1	0.1561	8123	25.1533	28.4390
			SA	1	0.0338	1155	17.7679	2.0795
			RQA-100	1	0.0096	449	8.2183	106.9570
OR5x250-0.50_10	5	250	HPSOGO	1	0.0748	6950	22.6577	29.0740
			SA	1	0.0195	1297	17.7065	2.5535
			RQA-100	1	0.0026	207	5.3474	122.8820
OR5x250-0.50_1	5	250	HPSOGO	1	0.0729	7171	20.3496	26.5765
			SA	1	0.0187	1291	18.0700	2.5690
			RQA-100	1	0.0031	253	6.5065	118.1420
OR5x250-0.50_2	5	250	HPSOGO	1	0.0747	7185	23.6992	25.8775
			SA	1	0.0180	1222	17.1683	2.5635
			RQA-100	1	0.0027	194	7.0813	116.9920
OR5x250-0.50_3	5	250	HPSOGO	1	0.0785	7026	22.9483	25.0560
			SA	1	0.0190	1256	20.4217	2.5700
			RQA-100	1	0.0017	104	5.5009	121.8500
OR5x250-0.50_4	5	250	HPSOGO	1	0.0772	7637	17.0112	25.1990
			SA	1	0.0182	1591	16.7562	2.5685
			RQA-100	1	0.0023	31	7.3086	127.1280
OR5x250-0.50_5	5	250	HPSOGO	1	0.0716	7203	22.6324	25.9030
			SA	1	0.0214	1994	16.5378	2.5595
			RQA-100	1	0.0025	141	6.9390	131.0560
OR5x250-0.50_6	5	250	HPSOGO	1	0.0737	7254	22.9772	30.5640
			SA	1	0.0180	1519	16.3213	2.5565
			RQA-100	1	0.0025	182	5.6732	126.0940
OR5x250-0.50_7	5	250	HPSOGO	1	0.0753	6279	22.6303	32.5725
			SA	1	0.0168	1208	19.0788	2.9430
			RQA-100	1	0.0025	202	6.3269	131.2820
OR5x250-0.50_8	5	250	HPSOGO	1	0.0725	6855	22.0673	29.9005
			SA	1	0.0176	983	14.8694	2.5530
			RQA-100	1	0.0031	241	6.3328	127.2250
OR5x250-0.50_9	5	250	HPSOGO	1	0.0762	7391	22.3987	29.6950
			SA	1	0.0198	1458	18.6615	2.5460
			RQA-100	1	0.0020	113	6.2984	123.9440
OR5x250-0.75_10	5	250	HPSOGO	1	0.0522	6433	25.8322	29.1210
			SA	1	0.0093	967	14.3776	2.7475
			RQA-100	1	0.0004	14	4.4283	115.7700
OR5x250-0.75_1	5	250	HPSOGO	1	0.0476	5152	23.7869	28.0415
			SA	1	0.0096	668	15.1189	2.3370
			RQA-100	1	0.0010	99	5.0666	115.1620

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR5x250-0.75_2	5	250	HPSOGO	1	0.0533	6507	28.6059	26.2205
			SA	1	0.0094	1016	12.9906	2.3685
			RQA-100	1	0.0008	60	5.3982	112.3330
OR5x250-0.75_3	5	250	HPSOGO	1	0.0491	6567	21.3401	26.1300
			SA	1	0.0088	821	16.8045	2.3340
			RQA-100	1	0.0014	131	4.6819	116.8470
OR5x250-0.75_4	5	250	HPSOGO	1	0.0502	6064	22.7420	26.6215
			SA	1	0.0095	1022	13.7244	2.3360
			RQA-100	1	0.0007	42	5.5353	114.6650
OR5x250-0.75_5	5	250	HPSOGO	1	0.0506	6541	22.7815	25.4620
			SA	1	0.0094	859	14.7665	2.6090
			RQA-100	1	0.0008	57	5.4130	113.7870
OR5x250-0.75_6	5	250	HPSOGO	1	0.0507	6585	21.8632	31.3410
			SA	1	0.0101	1031	15.6764	2.3255
			RQA-100	1	0.0012	121	4.8503	113.9140
OR5x250-0.75_7	5	250	HPSOGO	1	0.0515	6432	25.0708	30.4805
			SA	1	0.0103	982	16.5499	2.3635
			RQA-100	1	0.0009	26	5.3310	113.6910
OR5x250-0.75_8	5	250	HPSOGO	1	0.0528	6358	25.8175	28.3835
			SA	1	0.0081	717	15.2807	2.3230
			RQA-100	1	0.0007	55	4.8052	115.1330
OR5x250-0.75_9	5	250	HPSOGO	1	0.0517	7008	23.3843	27.2905
			SA	1	0.0097	1084	16.9770	2.3265
			RQA-100	1	0.0004	14	5.1303	113.4200
OR5x500-0.25_10	5	500	HPSOGO	1	0.2063	23069	31.2777	61.1155
			SA	1	0.0392	3332	23.4868	3.5885
			RQA-100	1	0.0113	1224	8.3833	159.3750
OR5x500-0.25_1	5	500	HPSOGO	1	0.1933	20744	30.7413	65.2450
			SA	1	0.0393	3290	20.3686	3.5780
			RQA-100	1	0.0125	1251	7.5551	158.0500
OR5x500-0.25_2	5	500	HPSOGO	1	0.1977	21488	31.6109	60.1515
			SA	1	0.0407	3952	23.7396	3.6160
			RQA-100	1	0.0122	1116	11.9352	164.3780
OR5x500-0.25_3	5	500	HPSOGO	1	0.1901	20535	27.9639	62.6610
			SA	1	0.0397	3307	21.9238	3.5760
			RQA-100	1	0.0114	854	11.2590	172.2380
OR5x500-0.25_4	5	500	HPSOGO	1	0.2001	22456	33.8615	56.1965
			SA	1	0.0399	3932	22.8265	3.5995
			RQA-100	1	0.0153	1458	9.8133	163.9920
OR5x500-0.25_5	5	500	HPSOGO	1	0.2048	22749	29.4433	61.2805
			SA	1	0.0379	3619	23.5716	3.5895
			RQA-100	1	0.0109	1126	9.6499	168.3920

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR5x500-0.25_6	5	500	HPSOGO	1	0.2147	24342	29.1794	52.9320
			SA	1	0.0373	3559	20.7014	3.5860
			RQA-100	1	0.0128	1268	9.5854	181.0770
OR5x500-0.25_7	5	500	HPSOGO	1	0.2035	20245	35.1998	50.6235
			SA	1	0.0396	3396	22.9116	3.5920
			RQA-100	1	0.0106	999	8.4012	169.9220
OR5x500-0.25_8	5	500	HPSOGO	1	0.2065	21192	33.9731	52.6590
			SA	1	0.0413	3927	20.6289	3.5780
			RQA-100	1	0.0114	1078	10.9567	159.0370
OR5x500-0.25_9	5	500	HPSOGO	1	0.1929	20578	34.9763	61.1925
			SA	1	0.0398	3620	26.9147	3.5920
			RQA-100	1	0.0133	1373	9.7581	151.9860
OR5x500-0.50_10	5	500	HPSOGO	1	0.0943	18718	24.3619	58.2825
			SA	1	0.0228	3624	22.8695	4.9105
			RQA-100	1	0.0037	698	7.8753	246.3870
OR5x500-0.50_1	5	500	HPSOGO	1	0.0915	17451	29.6742	55.8545
			SA	1	0.0221	3697	23.0257	4.6005
			RQA-100	1	0.0039	712	8.0312	247.6300
OR5x500-0.50_2	5	500	HPSOGO	1	0.0943	18743	25.3047	51.9705
			SA	1	0.0228	3502	24.8962	4.6085
			RQA-100	1	0.0039	683	7.3263	198.2870
OR5x500-0.50_3	5	500	HPSOGO	1	0.0946	16671	31.1525	51.5395
			SA	1	0.0222	4021	18.9953	4.6210
			RQA-100	1	0.0031	554	6.9732	231.1660
OR5x500-0.50_4	5	500	HPSOGO	1	0.0962	19475	27.8704	61.5290
			SA	1	0.0238	3773	28.7573	4.6200
			RQA-100	1	0.0026	472	6.1156	229.4700
OR5x500-0.50_5	5	500	HPSOGO	1	0.0960	19359	25.5096	53.5610
			SA	1	0.0261	4696	21.1764	4.6155
			RQA-100	1	0.0030	337	8.0433	205.6060
OR5x500-0.50_6	5	500	HPSOGO	1	0.0969	20034	24.6759	50.0180
			SA	1	0.0242	4262	20.6083	4.6430
			RQA-100	1	0.0026	478	6.7153	230.7250
OR5x500-0.50_7	5	500	HPSOGO	1	0.0969	19030	28.7074	49.2515
			SA	1	0.0231	3689	22.1376	4.6340
			RQA-100	1	0.0034	565	8.0137	224.8190
OR5x500-0.50_8	5	500	HPSOGO	1	0.0947	19012	23.9654	49.1385
			SA	1	0.0244	4385	21.4075	4.7535
			RQA-100	1	0.0034	521	9.2499	220.6110
OR5x500-0.50_9	5	500	HPSOGO	1	0.0948	19000	31.6606	49.2685
			SA	1	0.0248	3992	23.3645	4.6070
			RQA-100	1	0.0032	579	6.9771	220.1250

instance	C	N	solver	SR	MAPE	LE	SD	time [s]
OR5x500-0.75_10	5	500	HPSOGO	1	0.0727	19143	30.4953	48.9025
			SA	1	0.0107	2353	19.5588	4.0050
			RQA-100	1	0.0011	263	6.1380	220.9980
OR5x500-0.75_1	5	500	HPSOGO	1	0.0688	18341	26.5748	48.9000
			SA	1	0.0117	2227	22.2070	4.0475
			RQA-100	1	0.0014	345	4.8606	218.1030
OR5x500-0.75_2	5	500	HPSOGO	1	0.0699	19019	34.0746	48.8975
			SA	1	0.0115	2747	20.2237	4.0330
			RQA-100	1	0.0012	256	5.9632	216.1120
OR5x500-0.75_3	5	500	HPSOGO	1	0.0711	19234	27.4238	49.1380
			SA	1	0.0111	2530	19.5597	4.3235
			RQA-100	1	0.0013	254	6.0704	211.8480
OR5x500-0.75_4	5	500	HPSOGO	1	0.0694	18538	30.5531	48.8360
			SA	1	0.0114	2689	20.0734	4.0475
			RQA-100	1	0.0011	265	6.7030	218.5210
OR5x500-0.75_5	5	500	HPSOGO	1	0.0707	19396	28.3701	48.8975
			SA	1	0.0109	2398	19.7829	4.0440
			RQA-100	1	0.0012	307	5.2340	217.9460
OR5x500-0.75_6	5	500	HPSOGO	1	0.0717	19613	33.3697	49.1335
			SA	1	0.0112	2619	21.5855	4.0060
			RQA-100	1	0.0011	233	6.9699	209.6230
OR5x500-0.75_7	5	500	HPSOGO	1	0.0712	19765	27.7224	49.0875
			SA	1	0.0115	2430	19.8531	4.0055
			RQA-100	1	0.0013	301	5.2995	216.1530
OR5x500-0.75_8	5	500	HPSOGO	1	0.0705	19182	29.6816	49.8395
			SA	1	0.0127	2977	20.2320	4.0380
			RQA-100	1	0.0014	278	6.8800	206.2360
OR5x500-0.75_9	5	500	HPSOGO	1	0.0703	19787	26.2078	49.4645
			SA	1	0.0119	2759	20.3187	4.0115
			RQA-100	1	0.0011	239	5.4827	221.5800

Acknowledgements

Before all, I would like to thank my thesis advisor Prof. Andrea Acquaviva, for his helpfulness and care in directing my work and for facilitating my internship with NEC Japan.

I am also profoundly grateful to my thesis co-advisor Gianvito Urgese, who never failed to address my doubts and followed my work with enthusiasm and friendliness.

It was a great pleasure to work with my internship supervisor in NEC Japan, Yuki Kobayashi, and I thank him for his attentive and keen guidance and encouragement during my stay in Japan.

I also thank my colleague Marco for his friendship and for his invaluable assistance in navigating the joys and troubles of my internship.

To my family, who supported me not only financially but morally throughout my studies and afforded me the incredible privilege of pursuing two Bachelor's Degrees and one Master's Degree, goes my deepest gratitude and affection.

I would be remiss not to thank all the talented, smart, and kind colleagues that I have met throughout my years in Politecnico: your passion, dedication and strength inspired me every day of my studies.

Last and not least, thank you to the friends who cheered me on through the exams, the late nights, and the long months away from home: your support means the world to me.

Bibliography

[1] James C. T. Mao and B. A. Wallingford. "An Extension of Lawler and Bell's Method of Discrete Optimization with Examples from Capital Budgeting". In: *Management Science* 15.2 (1968), B51–B60. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/2628852.

[2] Jay D. Schwartz, Wenlin Wang, and Daniel E. Rivera. "Simulation-based optimization of process control policies for inventory management in supply chains". In: *Automatica* 42.8 (2006). Optimal Control Applications to Management Sciences, pp. 1311–1320. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2006.03.019. URL: http://www.sciencedirect.com/science/article/pii/S0005109806001464.

[3] Hanhong Zhu et al. "Particle Swarm Optimization (PSO) for the constrained portfolio optimization problem". In: *Expert Systems with Applications* 38.8 (2011), pp. 10161–10169. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2011.02.075. URL: http://www.sciencedirect.com/science/article/pii/S0957417411002818.

[4] Felix T.S. Chan and Rahul Swarnkar. "Ant colony optimization approach to a fuzzy goal programming model for a machine tool selection and operation allocation problem in an FMS". In: *Robotics and Computer-Integrated Manufacturing* 22.4 (2006), pp. 353–362. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2005.08.001. URL: http://www.sciencedirect.com/science/article/pii/S073658450500061X.

[5] Mir Saman Pishvaee, Fariborz Jolai, and Jafar Razmi. "A stochastic optimization model for integrated forward/reverse logistics network design". In: *Journal of Manufacturing Systems* 28.4 (2009), pp. 107–114. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2010.05.001. URL: http://www.sciencedirect.com/science/article/pii/S027861251000018X.

[6] Ali R. Guner and Mehmet Sevkli. "A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem". In: *J. Artif. Evol. App.* 2008 (Jan. 2008), 10:1–10:9. ISSN: 1687-6229. DOI: 10.1155/2008/861512. URL: http://dx.doi.org/10.1155/2008/861512.

[7] Brusco Michael J. and Jacobs Larry W. "A simulated annealing approach to the cyclic staff-scheduling problem". In: *Naval Research Logistics (NRL)* 40.1 (), pp. 69–84. DOI: 10.1002/1520-6750(199302)40:1<69::AID-NAV3220400105>3.0.CO;2-H. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/1520-6750%28199302%2940%3A1%3C69%3A%3AAID-NAV3220400105%3E3.0.CO%3B2-H. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/1520-6750%28199302%2940%3A1%3C69%3A%3AAID-NAV3220400105%3E3.0.CO%3B2-H.

[8] Jonathan F. Bard, David P. Morton, and Yong Min Wang. "Workforce planning at USPS mail processing and distribution centers using stochastic optimization". In: *Annals of Operations Research* 155.1 (Nov. 2007), pp. 51–78. ISSN: 1572-9338. DOI: 10.1007/s10479-007-0213-1. URL: https://doi.org/10.1007/s10479-007-0213-1.

[9] Clayton Commander. "A Survey of the Quadratic Assignment Problem, with Applications". In: 4 (Jan. 2005).

[10] *Optimization problem - Wikipedia, the free encyclopedia*. https://en.wikipedia.org/wiki/Optimization_problem.

[11] *Concorde Home*. http://www.math.uwaterloo.ca/tsp/concorde.html.

[12] *TSPLIB*. https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

[13] M.Grazia Speranza and Renata Mansini. "CORAL: An Exact Algorithm for the Multidimensional Knapsack Problem". In: 24 (July 2012), pp. 399–415.

[14] Zhao Xing and Weixiong Zhang. "MaxSolver: An efficient exact algorithm for (weighted) maximum satisfiability". In: *Artificial Intelligence* 164.1 (2005), pp. 47–80. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2005.01.004. URL: http://www.sciencedirect.com/science/article/pii/S0004370205000160.

[15] Beheshti and Shamsuddin. "A Review of Population-based Meta-Heuristic Algorithm". In: *Int. J. Advance. Soft Comput. Appl., Vol. 5, No. 1, March 2013* (2013).

[16] Scott Kirkpatrick, C. D. Gelatt, and Mario P. Vecchi. "Optimization by simulated annealing." In: *Science* 220 4598 (1983), pp. 671–80.

[17] Jenifer J, Anand S, and Levingstan Y. "SIMULATED ANNEALING ALGORITHM FOR MODERN VLSI FLOORPLANNING PROBLEM". In: 2 (Apr. 2016), pp. 175–181.

[18] Atanu Roy Karthik Ganesan Pillai. "Parallel Simulated Annealing for VLSI Cell Placement Problem". In: 2009.

[19] Adrian Ludwin and Vaughn Betz. "Efficient and Deterministic Parallel Placement for FPGAs". In: *ACM Trans. Design Autom. Electr. Syst.* 16 (2011), 22:1–22:23.

[20] Y. Han, S. Roy, and K. Chakraborty. "Optimizing simulated annealing on GPU: A case study with IC floorplanning". In: *2011 12th International Symposium on Quality Electronic Design*. Mar. 2011, pp. 1–7. DOI: 10.1109/ISQED.2011.5770735.

[21] Sanroku Tsukamoto et al. "An Accelerator Architecture for Combinatorial Optimization Problems". In: 2017.

[22] M. Baity-Jesi et al. "Janus II: A new generation application-driven computer for spin-system simulations". In: *Computer Physics Communications* 185.2 (2014), pp. 550–559. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2013.10.019. URL: http://www.sciencedirect.com/science/article/pii/S0010465513003470.

[23] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992. ISBN: 0262082136.

[24] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Neural Networks, 1995. Proceedings., IEEE International Conference on*. Vol. 4. Nov. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[25] Tadashi Kadowaki and Hidetoshi Nishimori. "Quantum Annealing in the Transverse Ising Model". In: 58 (Apr. 1998).

[26] Roman Marto ňák, Giuseppe E. Santoro, and Erio Tosatti. "Quantum annealing by the path-integral Monte Carlo method: The two-dimensional random Ising model". In: *Phys. Rev. B* 66 (9 Sept. 2002), p. 094203. DOI: 10.1103/PhysRevB.66.094203. URL: https://link.aps.org/doi/10.1103/PhysRevB.66.094203.

[27] Bettina Heim et al. "Quantum versus classical annealing of Ising spin glasses". In: *Science* 348.6231 (2015), pp. 215–217. ISSN: 0036-8075. DOI: 10.1126/science.aaa4170. eprint: http://science.sciencemag.org/content/348/6231/215.full.pdf. URL: http://science.sciencemag.org/content/348/6231/215.

[28] Arnab Das, Bikas K. Chakrabarti, and Robin B. Stinchcombe. "Quantum annealing in a kinetically constrained system". In: *Phys. Rev. E* 72 (2 Aug. 2005), p. 026701. DOI: 10.1103/PhysRevE.72.026701. URL: https://link.aps.org/doi/10.1103/PhysRevE.72.026701.

[29] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* 2 (2014), p. 5. ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005. URL: https://www.frontiersin.org/article/10.3389/fphy.2014.00005.

[30] F. Ortega-Zamorano et al. "FPGA Hardware Acceleration of Monte Carlo Simulations for the Ising Model". In: *IEEE Transactions on Parallel and Distributed Systems* 27.9 (Sept. 2016), pp. 2618–2627. ISSN: 1045-9219. DOI: 10.1109/TPDS.2015.2505725.

[31] Davide Venturelli et al. "Quantum Optimization of Fully Connected Spin Glasses". In: *Phys. Rev. X* 5 (3 Sept. 2015), p. 031040. DOI: 10.1103/PhysRevX.5.031040. URL: https://link.aps.org/doi/10.1103/PhysRevX.5.031040.

[32] P. Bergé et al. "Restricting the search space to boost Quantum Annealing performance". In: *2016 IEEE Congress on Evolutionary Computation (CEC)*. July 2016, pp. 3238–3245. DOI: 10.1109/CEC.2016.7744199.

[33] N. Kumaraguruparan, H. Sivaramakrishnan, and S. S. Sapatnekar. "Residential task scheduling under dynamic pricing using the multiple knapsack method". In: *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*. Jan. 2012, pp. 1–6. DOI: 10.1109/ISGT.2012.6175656.

[34] R. Camati, A. Calsavara, and Jr L.L. "Solving the virtual machine placement problem as a multiple multidimensional knapsack problem". In: *Proceedings of ICN 2014, The Thirteenth*. 2014. URL: http://www.thinkmind.org/index.php?View=article&articleid=icn_2014_11_10_30065.

[35] R. Mansini and M. G. Speranza. "A Multidimensional Knapsack Model for Asset-Backed Securitization". In: *The Journal of the Operational Research Society* 53.8 (2002), pp. 822–832. ISSN: 01605682, 14769360. URL: http://www.jstor.org/stable/822910.

[36] D. C. Vanderster, N. J. Dimopoulos, and R. J. Sobie. "Metascheduling Multiple Resource Types using the MMKP". In: *2006 7th IEEE/ACM International Conference on Grid Computing*. Sept. 2006, pp. 231–237. DOI: 10.1109/ICGRID.2006.311020.

[37] Madjid Tavana, Kaveh Khalili-Damghani, and Amir-Reza Abtahi. "A fuzzy multidimensional multiple-choice knapsack model for project portfolio selection using an evolutionary algorithm". In: *Annals of Operations Research* 206.1 (July 2013), pp. 449–483. ISSN: 1572-9338. DOI: 10.1007/s10479-013-1387-3. URL: https://doi.org/10.1007/s10479-013-1387-3.

[38] Mohammed El-Shafei, Imtiaz Ahmad, and Mohammad Gh. Alfailakawi. "Hardware accelerator for solving 0-1 knapsack problems using binary harmony search". In: *International Journal of Parallel, Emergent and Distributed Systems* 33.1 (2018), pp. 87–102. DOI: 10.1080/17445760.2017.1324025. eprint: https://doi.org/10.1080/17445760.2017.1324025. URL: https://doi.org/10.1080/17445760.2017.1324025.

[39] Vincent Boyer, Didier El Baz, and Moussa Elkihel. "Solving knapsack problems on GPU". In: *Computers and Operations Research* 39.1 (2012), pp. 42–47. DOI: 10.1016/j.cor.2011.03.014. URL: https://hal.archives-ouvertes.fr/hal-01152223.

[40] Luis Fernando Mingo López, Nuria Gómez Blas, and Alberto Arteta Albert. "Multidimensional knapsack problem optimization using a binary particle swarm model with genetic operations". In: *Soft Computing* 22.8 (Apr. 2018), pp. 2567–2582. ISSN: 1433-7479. DOI: 10.1007/s00500-017-2511-0. URL: https://doi.org/10.1007/s00500-017-2511-0.

[41] Kazutoshi Wakabayashi. "CyberWorkBench: Integrated design environment based on C-based behavior synthesis and verification". In: *VLSI Design, Automation and Test, 2005.(VLSI-TSA-DAT). 2005 IEEE VLSI-TSA International Symposium on*. IEEE. 2005, pp. 173–176.

[42] Intel - Altera. *Avalon Interface Specifications*. URL: https://www.altera.com/documentation/nik1412467993397.html (visited on 06/29/2018).

[43] *Zuse Institute Berlin - MP-Testdata - SAC-94 Multiple-Knapsack Problems*. http://elib.zib.de/pub/mp-testdata/ip/sac94-suite/index.html.

[44] H. Martin Weingartner and David N. Ness. "Methods for the Solution of the Multidimensional 0/1 Knapsack Problem". In: *Operations Research* 15.1 (1967), pp. 83–103. DOI: 10.1287/opre.15.1.83. eprint: https://doi.org/10.1287/opre.15.1.83. URL: https://doi.org/10.1287/opre.15.1.83.

[45] P.C. Chu and J.E. Beasley. "A Genetic Algorithm for the Multidimensional Knapsack Problem". In: *Journal of Heuristics* 4.1 (June 1998), pp. 63–86. ISSN: 1572-9397. DOI: 10.1023/A:1009642405419. URL: https://doi.org/10.1023/A:1009642405419.

[46] Intel - Altera. *Altera Stratix V FPGA*. URL: https://www.altera.com/products/fpga/stratix-series/stratix-v/features.html (visited on 02/25/2017).

[47] A. Boukedjar, M. E. Lalami, and D. El-Baz. "Parallel Branch and Bound on a CPU-GPU System". In: *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*. Feb. 2012, pp. 392–398. DOI: 10.1109/PDP.2012.23.

[48] Intel. *Intel Core i7 Processor Series Datasheet, Vol. 1.* URL: https://www.intel.com/content/www/us/en/processors/core/core-i7-900-ee-and-desktop-processor-series-datasheet-vol-1.html (visited on 06/29/2018).

[49] *OR-Library.* http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html.