1QBit

Simulated Annealing via Quantum Annealing

A General Framework for Continuous Optimization

White Paper

Simulated Annealing via Quantum Annealing

A General Framework for Continuous Optimization

Pooya Ronagh

Abstract

In the summer of 2013, 1QBit developed a general method for solving continuous optimization problems. This method works under the assumption of the existence of a computation model with a Turing reduction of problems to either quadratic unconstrained binary optimization (QUBO) problems or to an Ising spin problem. We assume the existence of a Turing machine with an oracle that answers the result of an optimization of one of the above families of problems in one clock of the machine. This investigation is motivated by the assumption that the processor produced by D-Wave Systems, Inc. is a successful realization of such an oracle.

Our method is inspired by different types of simulated annealing and genetic algorithms. Below we explain a general framework for tackling optimization problems using this new method. Similar to the case of simulated annealing, there are several ways of implementing different submodules of our algorithm. Our implementation provides several famous such submodules and can receive them as user inputs. These can be chosen depending on the optimization problem at hand and convenience to the user.

This paper presents an application of this method to a mixed-integer optimization problem. This will demonstrate an interesting method of representing a cardinality-constrained optimization problem using analytic expressions, and the ability of the method to solve such mixed-integer optimization problems.
Keywords: random search algorithms, stochastic optimization methods, simulated annealing, adiabatic quantum computing, D-Wave System

1 Introduction

In this section we give a brief introduction to the theory of simulated annealing. An extensive overview and survey of results can be found in [8].

Simulated annealing is a general optimization technique that has been used for the past 40 years in various applications. Originally [7], it was thought of as a method of solving combinatorial optimization problems (e.g., Travelling Salesperson and VLSI design), where the objective function is defined on a domain with discrete topology. Mathematical models of

©2016 1QB Information Technologies

this algorithm are essentially Markov chain stochastic processes, and prove probabilistic convergence results of the algorithm to the global optimal answers of the optimization problem under certain assumptions (c.f., for example, [4]).

Various key features of the algorithm, for instance the definition of some type of neighbourhood structure (e.g., a topology, a metric, or an adjacency graph) on the feasible region, a cooling schedule, generation of points, acceptance criteria, population sizes, etc. are general concepts that need to be selected accordingly for the specific application one is tackling. The performance of the algorithm depends heavily on a wise selection of such features. Theoretical results (such as proof of asymptotic convergence) also depend on the specific selection of these features. [11, 4, 1, 9]

Later on (in [12] for the first time), the concept of simulated annealing was applied and benchmarked for continuous optimization problems. The continuous method shows promising results in several cases of non-convex and mixed-integer optimization problems. Our proposal is a variation of the continuous simulated annealing technique suggested for these problem types. Our algorithm intrinsically works with populations of points rather than a single random process as it takes advantage of the QUBO/Ising solver oracle. An advantage to the D-Wave system is that it can produce a spectrum of optimal and suboptimal answers to a QUBO/Ising optimization problem, rather than merely the lowest energy point. We hence use ideas of population-oriented simulated annealing algorithms to most efficiently benefit from the spectrum of received answers. Our claim is that this algorithm shows an improvement in computation time.

2 A non-technical overview of simulated annealing

Simulated annealing is a search method that explores a search space randomly in order to find a desired object. One can think of this technique as analogous to the way molecules in a metal cool down into a crystalline structure. At higher temperatures the molecules are in excited states and are in unorganized positions with respect to each other. As the metal is slowly annealed the molecules start to align themselves in a low-energy arrangement, one we call crystalline.

One can think of the algorithm as an *agent* searching the space. By "bouncing" from one location to another, the agent finds the point with the lowest altitude in their universe. The trajectory of such an agent is called a *random process*. If the agent has a low memory span, it will "forget" the path it took before reaching its current location. The next bounce depends only on the current placement, not previous bounces. In mathematics, such a random process is called a *Markov chain*. The algorithm might have several such agents that communicate the information they find about the search space with each other. In this case, the overall algorithm bounces from one *population* of locations to another.

The key point about simulated annealing is the notion of *temperature*. At the beginning of their bouncing, the agents are more excited. As a result, they occasionally bounce uphill rather than going downhill. This may seem like a bad strategy to find a minimum, but it actually helps agents to avoid settling in local small valleys, and reach deeper valleys. Bouncing uphill happens according to an *acceptance criteria*. The acceptance criteria is satisfied if a certain random number is in a given range. As the temperature decays according to a *temperature schedule* or *cooling schedule* the agents lose their respective energy and the bounces become smaller. In other words, the acceptance criteria becomes harder to satisfy when the world becomes frozen.

Our algorithm is a very specific modification of typical simulated annealing algorithms. In our algorithm, the agents consider jumping to an exponential number of new points. This exponential collection of points is chosen carefully so that the altitude of each point can be compared using the D-Wave system and ranked. The agents then decide to bounce to a new altitude according to both their level of excitation and this ranking.

An underlying hypothesis is that this comparison of an exponential number of altitudes can be performed in less than exponential time using the D-Wave system. One of the biggest speed bottlenecks of simulated annealing algorithms is the evaluation of these altitudes, which we are hereby minimizing to an exponentially low fraction of time. Therefore, the claim is that our algorithm will show significant improvement in speed compared with other simulated annealing procedures when the dimension of the search space is high.

©2016 1QB Information Technologies

3 The concept of simulated annealing

In this section, we list several components of a simulated annealing algorithm. Let D be a set and $y = f(x)$ a real-valued function on D. The most general form of the optimization problem we want to solve can be simply phrased as

$$\text{minimize} \quad y = f(x) \quad x \in D\,, \tag{1}$$
$$\text{subject to} \quad x \in F. \tag{2}$$

The function $f(x)$ is called the *objective function* and the set D is called the *domain of definition*. The subset $F \subseteq D$ is called the *feasible region*. A simulated annealing procedure searches among the points $x \in D$ in order to find an optimal solution $x^* \in F$ for the optimization problem. Depending on the specific problem, the algorithm might be searching only points that are included in F or more generally be exploring all of D (rejecting points in $D \setminus F$ by means of penalty functions, etc.). It is also possible that the two sets F and D coincide. We blur this distinction and use a third term, *search space*, and abuse the notation D for it when the discussion applies to either or both of F and D.

It is beneficial for a simulated annealing algorithm if the problem at hand has some further structure:

3.1 Neighbourhood structure

We need D to have some *neighbourhood structure*, denoted by $\mathcal{N}$, that gives us a gauge of points being close or far from each other. Some examples of such structures are topologies and metrics, and in case D has the structure of a vector space, $\mathcal{N}$ can be a norm or a semi-norm on D. For theoretical analysis of a simulated annealing algorithm one needs D to be equipped with a measure. We may also take advantage of f respecting $\mathcal{N}$. For instance, if $\mathcal{N}$ is a topology, then we can require for $f(x)$ to be continuous on D. If $\mathcal{N}$ is a metric, we may require $f(x)$ to be continuous or differentiable to some order on D.

Example 1. In fact, most practical optimization problems in optimization theory are stated more explicitly as

$$\begin{aligned} \text{minimize} \quad & y = f(x) \quad x \in \mathbb{R}^N\,, \\ \text{subject to} \quad & g_i(x) = a_i \text{ for all } i = 1, \cdots, r, \\ & h_i(x) \le b_i \text{ for all } i = 1, \cdots, s. \end{aligned} \tag{3}$$

In this example $D = \mathbb{R}^N$ and $y = f(x)$ is a function of N real variables. The feasible region F is a closed manifold (possibly with a non-empty boundary) embedded in $\mathbb{R}^N$ cut out by the equality and inequality constraints above, and $\mathcal{N}$ is the induced metric. The objective function and the equality and inequality constraints may be continuous, differential, or smooth. The constraints can also impose mixed-integer constraints on the feasible region (in particular the manifold F does not need to be connected).

3.2 Temperature schedule

A simulated annealing procedure depends on a sequence of positive real parameters $\{\tau_t\}_{t=0}^{\infty}$ called *temperature* (a.k.a. *control parameter* in many references) that is converging to zero through iteration of the algorithm on the index t. Physics' interpretation of this convergence is that the system is approaching a "frozen" state with zero entropy, in which case all random processes involved in the algorithm become trivial. At each temperature τ_t, the algorithm simulates a random process (usually a discrete-time Markov chain) on D or in case of multi-agent systems on a finite cartesian product $D \times \cdots \times D$ (c.f. §3.6). In practice these Markov chains are of finite length ℓ^{τ_t}. The initial distribution of the Markov chain depends on the final state of the algorithm at previous temperature τ_{t-1}. A temperature schedule consists of,

1. an initial value τ_0 for the temperature schedule;
2. a recursive expression for generating a next temperature, τ_t, from previous ones $(\tau_0, \cdots, \tau_{t-1})$ such that $\lim_{t\to\infty} \tau_t = 0$;
3. a length ℓ^{τ_t}, for the Markov chain on D generated at a given temperature τ_t;
4. a *stopping criterion* for terminating the truncating of the sequence $\{\tau_t\}_t$ in finite length, in which case the algorithm terminates.

3.3 Transition probability

Given any point $x \in D$, at a given temperature $\tau = \tau_t$, we need a probability density function $\mu = \mu_x^\tau$ on D. This distribution will eventually define the transition kernel of the Markov chain described above from point x to point y at temperature τ of the algorithm:

$$K^\tau(x,y) = \alpha^\tau(x,y)\,\mu_x^\tau(y)\,.$$

One may think of μ_x^τ as a means of generating a new point y from a given point x and therefore we use the term *generating distribution* centred at $x \in D$ for it. For theoretical considerations we need μ to be a continuous function $\mu^\tau : D \times D \to \mathbb{R}$ (for instance, with respect to the Lebesgue measure on D). And we assume $\mu_x^\tau := \mu^\tau(x,.)$ to be a probability density function on D and $\mu^\tau(.,y)$ to be measurable.

The neighbourhood structure on D often makes its presence in the definition of μ_x^τ.

Example 2. Let $D = \mathbb{R}^n$. We may define μ_x^τ to be the n-dimensional Gaussian distribution with mean x and variance $\frac{\tau}{\sqrt{2}}$ as in the case of classical simulated annealing (a.k.a. Boltzmann machines) [11]. Notice that since $\lim_t \tau_t = 0$, then these transition probabilities approach Dirac delta functions.

The important factor for definition of the transition kernel is that the Markov chain defined by it needs to be irreducible. When $\alpha(.,.)$ is the Boltzmann acceptance criteria as explained below, any of the following conditions is sufficient for the resulting Markov chain to be irreducible:

1. $\mu : D \times D \to \mathbb{R}$ is positive; or,

2. For every $\varepsilon > 0$, there exists $\delta > 0$ such that $\mu(x,y) > \varepsilon$ is $|x-y| < \delta$.

Example 3. Let $B_\varepsilon(x)$ denote the ball of radius ε centred at x:

$$B_\varepsilon(x) = \{y \in D : \text{dist}(x,y) \le \varepsilon\}.$$

If D is a bounded open subset of $\mathbb{R}^n$ with boundary $\partial D = \bar{D} \setminus D$, we may let $r = \text{dist}(x, \partial D)$ and define the transition probability to be the truncated Gaussian distribution with mean x in B_r. If D is a compact subset of $\mathbb{R}^n$, we may use the same distribution as above for interior points $x \in D^\circ$ and of $x \in \partial D$ we may set μ_x^τ to be the uniform distribution on D°.

3.4 Acceptance criteria

Given a temperature $\tau = \tau_t$, and two points $x_1, x_2 \in D$, the acceptance criteria $\alpha^\tau(x_1, x_2)$, determines the probability at which the point x_2 is accepted as the successor of x_1 in any of the random processes taking place in the algorithm at temperature τ. In essentially all implementations of simulated annealing, this acceptance criterion is inspired by physics to resemble the Boltzmann distribution:

Example 4. We recall the Boltzmann distribution (a.k.a. the Gibbs measure) in statistical mechanics. Given a continuous function $f : D \to \mathbb{R}$ and a free parameter β, this distribution is

$$p(x) = \frac{1}{Z(\beta)} e^{-\beta f(x)}\,,$$

where $Z(\beta) = \int_D p(x)$ is uniquely determined by f and β to be the normalizing constant (a.k.a. partition function) that turns the above expression into a probability density function.[6]

Given points x_1 and x_2 in D at temperature τ, we define the probability of acceptance of x_2 as successor of x_1 to be,

$$\alpha^\tau(x_1, x_2) = \begin{cases} 1 & \text{if } f(x_2) \le f(x_1)\,, \\ e^{-(f(x_2)-f(x_1))/\tau} \frac{\mu^\tau(x_2,x_1)}{\mu^\tau(x_1,x_2)} & \text{otherwise.} \end{cases}$$

The core theoretical proposition of simulated annealing is that

Theorem 1. Any irreducible Markov chain defined with the following kernel

$$K^\tau(x,y) = \alpha(x,y)\mu^\tau(x,y) + (1 - \int_D \alpha(x,z)\mu^\tau(x,z)dz)\delta(x-y)\,,$$

©2016 1QB Information Technologies

has stationary distribution $p(x)$. Here δ is the Dirac delta function.

Example 5. In this case the distribution of finding a random walk on D at temperature τ, to be at state x approaches the Boltzmann distribution:

$$\lim_{k\to\infty} P(X = x) = \frac{1}{Z} e^{-f(x)/\tau}.$$

Let X be a Markov chain on D with transition kernel

$$P(X_k = x_k | X_{k-1} = x_{k-1}) = K^\tau(x_{k-1}, x_k)\,.$$

The above acceptance criteria induces a new random process Y, with the following transition matrix:

$$P(Y_k = y_k | Y_{k-1} = y_{k-1}) = \begin{cases} K^\tau(x_{k-1}, x_k) A^\tau(x_{k-1}, x_k) & \text{if } x_k \neq x_{k-1}\,, \\ 1 - \sum_{y \neq x_{k-1}} K^\tau(x_{k-1}, y) A^\tau(x_{k-1}, y) & \text{otherwise.} \end{cases}$$

3.5 The algorithm

Simulated annealing is the following procedure:

```
inputs objective_function , temperature_schedule
initialize temperature_schedule
while stopping criterion not met do
    initialize temperature $\tau = \tau_0$
    initialize random process $X^\tau = (X_k^\tau)_{k=0}^\infty$ on $D$ and let $Y_0^\tau = X_0^\tau$
    while length of $(X_k^\tau)$ is less than $\ell^\tau$ do
        choose $Y_{k+1}^\tau$ from a distribution $\mu_{X_k}^\tau$
        if acceptance criterion is met according to $A^\tau(X_k^\tau, Y_{k+1}^\tau)$ then
            $X_{k+1}^\tau \leftarrow Y_{k+1}^\tau$
        else
            $X_{k+1}^\tau \leftarrow X_k^\tau$
        end if
    end while
    update temperature $\tau$
end while
return the minimum-valued point in the set $\{Y_k^\tau\}_{\tau,k}$
```

Mathematical analysis of a simulated annealing algorithm is generally a hard theoretical problem and even harder in the continuous case. For some nice convergence results of this sort we refer the reader to [1].

3.6 Multi-agent simulated annealing

One immediate generalization of the above algorithm is to let the algorithm simulate more than a single random process at each temperature. In this case, the algorithm has yet another component called the *population* function, $\pi = \pi^\tau$, which is a function of temperature τ. At given temperature $\tau = \tau_k$, the random process X^τ now is a tuple of π^τ random processes $X^{\tau,1}, \cdots, X^{\tau,\pi_\tau}$ on D, which in theory may be dependent or independent from each other. The distributions μ^τ and acceptance criteria A^τ are now more generally tuples of such functions as:

$$\mu = \mu^\tau_{X_k^1, \cdots, X_k^\pi}\,, \quad \text{and} \quad A^\tau(\{X_k^\tau\}_{k=1,\cdots,\pi}, Y_{k+1}^\tau)\,.$$

4 Inputs of the software

4.1 Conventional inputs

Optimization problem. The user specifies an optimization problem in the sense of 3. Such an object is an instance of a type which we denote as `objective_function`. More specifications of this object are provided in the documentation of the software. In particular, an instance of this input consists of an objective function, together with equality and inequality constraints.

©2016 1QB Information Technologies

Temperature schedule. Our software is able to operate with a user-defined temperature schedule. Users can also choose a temperature schedule from the library of various well known temperature schedules we have implemented (e.g., that of classical simulated annealing, fast simulated annealing, and very fast simulated re-annealing).

Distributions centred at each point. These functions can also be defined by the user according to the specific geometry of the optimization problem. Our default method is explained in §5.1.

Population. In the sense of §3.6, users may specify the population function for agents searching the feasible region of the optimization problem.

Local search method. After the simulated annealing algorithm terminates, we do a local search in the neighbourhood of the best solution found, in order to find the closest local minimum of the function. This can be done using any of the algorithms we have implemented (e.g., Monte Carlo, steepest descent, or conjugate gradient method). These algorithms require a specified precision that is also set as a user input.

4.2 New inputs

Unlike the above components, the core idea of our algorithm is a very specific definition of random processes X^τ and distributions μ_X^τ that makes exploitation of the power of the oracle possible for us. This will become clearer as we explain our algorithm in further detail below.

Population of candidates. In each round of use of the oracle we receive a spectrum of optimal and suboptimal points in the neighbourhood of a point $x \in D$. These points are treated as candidates for the next state of the Markov chain. This user-defined integer will determine the maximum size of the set of minimum-valued points that will be taken into consideration as candidates of transition from x to the next state.

Selection scheme. This is a generalization of acceptance criteria in the conventional simulated annealing algorithms. As explained above, our algorithm creates a population of candidate points in a neighbourhood of the current state of the Markov chain. The process of accepting a next proposed state x_k from the current x_{k-1} is now replaced by a *selection scheme* of x_k from a set of candidate points as a successor of x_{k-1}.

Users can select any of our built-in selection schemes (i.e., Boltzmann tournament scheme, Boltzmann scheme, and Cauchy scheme [3]) or define their favourite ones.

Before we introduce the last input parameter, recall that our software is a general-purpose simulated annealing optimizer that uses an oracle to solve QUBO or Ising model problems, in one instance being the D-Wave system. Let v be the largest number of variables in the problems we can solve with the oracle.

Search dimension. This parameter is a positive integer d, satisfying $1 \leq d \leq \min\{v, n\}$ where n is the dimension of the feasible region F. For example, if the equality constraints of 3 are all linearly independent, then the dimension of F is bounded by $n \leq N - r$ and consequently $d \leq \min\{v, N - r\}$. The role of this parameter will become clear in following sections. The performance of our algorithm heavily relies on choosing as large a parameter as possible.

5 Auxiliary components of the main algorithm

5.1 Approximately uniform distribution in a domain

We have implemented methods for generating random points in a given subset S of $\mathbb{R}^N$ (which can be the domain or the feasible region, for instance). The goal is for the distribution of such random points to be as close to the uniform distribution on S as possible. The effectiveness of a given method depends on the nature of the optimization problem at hand and the structure of S. Here are a few instances:

Example 6 (Standard euclidean box). Suppose $a_1, \cdots, a_n$ and $b_1, \cdots, b_n$ are real numbers such that $a_i \leq b_i$ for all $i = 1, \cdots, n$. Then for $S = [a_1, b_1] \times \cdots \times [a_n, b_n] \subseteq \mathbb{R}^n$, we select c_i from the uniform distribution on the interval $[a_i, b_i]$. Then the point $(c_1, \cdots, c_n)$ has uniform distribution on S.

Example 7 (Standard Euclidean simplex). Suppose S is the set of all points $(x_1, \cdots, x_{n+1}) \in \mathbb{R}^{n+1}$ satisfying $\sum_i x_i = 1$.

We use the method of Rubinstein and Melamed, for instance, to create a uniform distribution on S.[10]

Example 8 (Polytopes). If S is a polytope, we may use a hit-and-run Markov chain that starts from the Chebychev centre of the polytope and makes a random walk on the polytope to generate a random point within it in an approximately uniform way. We assign that as the centre of a hypercube. We then create an invertible matrix A uniformly randomly, that serves as the change of basis that determines the hypercube uniquely.[5]

The uniform distribution generated is used mainly as the initial distribution of Markov chains in simulated annealing. Its other applications are in creating generating distributions in the neighborhood of a point or when the generating distribution needs to be uniform (for example when the current state of the Markov chain is close to a boundary of the domain, c.f. example 3).

5.2 Transition to neighbour points

Given a point $c \in \mathbb{R}^n$ (generated as explained in the previous section) we now construct an affine transformation $\varphi(x) = Ax + b$ where A is a full-rank $N \times d$ matrix A and a column N-vector b. The affine transformation $\varphi(x) : \mathbb{R}^d \to \mathbb{R}^N$ will be such that either,

1. $\varphi(0) = c$ and $\varphi(\delta)$ is a point in S for all $\delta \in \{-1, 1\}^{\times d}$; or,
2. $\varphi(1/2, \cdots, 1/2) = c$ and $\varphi(\delta)$ is a point in S for all $\delta \in \{0, 1\}^{\times d}$.

Geometrically, this provides us with a hypercube of dimension d, with centre c centred at c and contained in a polytope contained in S (in practical examples such that 3, S and the feasible region coincide).

Given the generating distribution as in §3.3, we pick $v_1, \cdots, v_d$ in neighbourhood of the current state $x \in D$ according to it. The vectors $b_i = v_i - x$, form a family $\{b_1, \cdots, b_d\}$ and we require them to be linearly independent. These vectors produce an $N \times d$ matrix M.

We may need to dilate each vector b_i according to a scalar r_i. The goal is that after dilation all binary points $\delta \in \{-1, 1\}^{\times d}$ satisfy:

$$M\delta + c \in S\,, \quad \text{for all } \delta \in \{-1, 1\}^{\times d}\,.$$

It is easy to prove that construction of such a matrix M can be performed in time $\mathcal{O}(dN)$. The final result is that from a stochastically selected point $c \in S$, we now have the choice to transition to 2^d points in neighbourhood of c contained in S.

The matrix M above and centre point c, uniquely determine such pairs (A, b) of affine transformations from $\mathbb{R}^d$ to $\mathbb{R}^N$ satisfying the above conditions. In fact:

1. in the first case, we let $A = M$ and $b = c$; and
2. in the second case, we set $A = 1/2M$ and $b = c + 1/2M\mathbf{1}$, where **1** is the column N-vector of ones.

Example 9. Suppose S is the feasible region of the optimization problem 3 with linear constraints. This means that in addition to needing the hypercube to stay inside a polytope (the one cut out by the inequality constraints), the hypercube needs to reside in an affine hyperplane in $\mathbb{R}^N$ (the one cut out by the equality constraints). In this case, we select $b_1, \cdots, b_d$ in the kernel of the system of linear equalities. In case of the example in section 7 we also use a heuristic in choosing these vectors in a way such that the random processes do not converge to points on the boundaries of S.

5.3 Local polynomial approximation of objective function

Let f be the objective function $y = f(x)$ defined on the set S and $c \in S$ be a given point. Our goal is to approximate f with a polynomial in some neighbourhood of c; we will denote the result of this procedure as $P_{f,c}(x)$ for later reference. We provide various ways of performing this that can be selected by a user.

The first obvious way is Taylor approximation, $P_{f,c} = T_{f,c}(x)$, of $f(x)$ around centre c of the desired degree. The degree is chosen by the software as a function of modularity of the objective function, or it can be user-defined. However, Taylor approximations of high-degree are usually hard to compute, but in case the objective function has an analytic expression, we will need to find derivatives of it only once symbolically, and whenever a point $c \in S$ is given we

©2016 1QB Information Technologies

substitute c.

Other options include approximation using Legendre polynomials, Chebyshev polynomials, interpolation, and least square method.

5.4 Comparison of value of the neighbour points

Let $g(\delta) = P_{f,c}(A\delta + b)$ where $P_{f,c}$ is the polynomial generated in §5.3 and (A, b) is the affine transformation generated in §5.2. Thus, either:

- $g(\delta)$ is a polynomial in several spin variables (i.e. $\delta \in \{\pm 1\}^{\times n}$) and all points $A\delta + b$ are in S; or,
- $g(\delta)$ is a function in boolean variables (i.e. $\delta \in \{0, 1\}^{\times n}$) and all points $A\delta + b$ are in S.

The function g is now representative of an unconstrained polynomial optimization problem in variables $\{-1, 1\}$ or in boolean variables $\{0, 1\}$. We use the fact that every such optimization problem can be reduced to one in which the objective function is quadratic (but has auxiliary variables). This is done using a built-in reduction algorithm or can be user-defined.

We can now assume that the function g is representative of an unconstrained Ising spin model problem in case (1) or a quadratic unconstrained binary optimization problem (QUBO) in case (2). We use our oracle to solve the problem of optimization of g on a set of all binary points. The result is then interpreted by reversing the affine transformation above to a population P of candidate points. (In case of multi-agent systems in the sense of §3.6 the candidate population is a union of such sets $P = \cup_{i=1}^{\pi} P_i$.) We then generate our next state (or states, in case of multi-agent systems) from P according to the selection scheme explained in §4.2.

6 Main algorithm

Our main algorithm can be sketched using the following pseudo-code:

```
inputs objective_function, temperature_schedule
initialize temperature_schedule
while stopping criterion not met do
    initialize temperature $\tau = \tau_0$
    initialize random process $X^\tau = (X_k^\tau)_{k=0}^\infty$ as a tuple of $\pi^\tau$, Markov chains
        on $D$ and let $Y_0^\tau = X_0^\tau$ by random generation of points as in §5.1
    while length of $(X_k^\tau)$ is less than $\ell^\tau$ do
        determine $\pi^\tau \times 2^d$ neighbour points according to §5.2
        generate $\pi^\tau$ optimization problems in variables $\{-1, 1\}$ and $\{0, 1\}$ as in §5.4
        reduce these optimization problems to QUBOs according to §5.4
        solve these problems via oracle, and receive a list of (sub)-optimal solutions
        choose $(X_{k+1}^\tau)$ according to selection scheme §4.2
    end while
    update temperature $\tau$ according to temperature schedule §4.1
end while
return the minimum-valued point in the set $\{Y_k^\tau\}_{\tau,k}$
```

7 Case study: Cardinality constrained problems

Given a real vector $x = (x_1, \cdots, x_n) \in \mathbb{R}^n$ we define the support of x as the set of elements

$$\text{supp}(x) = \{i : x_i \neq 0\} \subseteq \{1, \cdots, n\}.$$

©2016 1QB Information Technologies

The key idea is that given a real vector $x \in \mathbb{R}^n$, the p-th power of the p-norm of x approaches the cardinality of supp(x):

$$\lim_{p \to 0} \|x\|_p^p = \lim_{p \to 0} \sum_{i=0}^{n} x_i^p = \#\,\text{supp}(x)\,.$$

Given an optimization problem 3 with an additional cardinality constraint

$$\begin{aligned} \text{minimize} \quad & y = f(x) \quad x \in \mathbb{R}^n\,, \\ \text{subject to} \quad & g_i(x) = a_i \text{ for all } i = 1, \cdots, r, \\ & h_i(x) \le b_i \text{ for all } i = 1, \cdots, s\,, \\ & \text{supp}(x) \le K. \end{aligned} \tag{4}$$

We may remove the cardinality constraint by adding a penalty term to the objective function

$$\begin{aligned} \text{minimize} \quad & y = f(x) + C\left(\|x\|_p^p - K\right)^2 \quad x \in \mathbb{R}^n\,, \\ \text{subject to} \quad & g_i(x) = a_i \text{ for all } i = 1, \cdots, r, \\ & h_i(x) \le b_i \text{ for all } i = 1, \cdots, s\,. \end{aligned} \tag{5}$$

where p is a small number in $(0, 1)$. This additional penalty term associates lower energy to sparser points $x \in \mathbb{R}^n$.

Remark 1. The lower p is, the better the cardinality constraint is approximated with this objective function. On the other hand, the Taylor expansions of the objective function are worse approximations of the function $f(x) + C\left(\|x\|_p^p - K\right)^2$ when p is closer to 0. Therefore, we empirically choose intermediate values for p.

Remark 2. As p ranges from 1 to 0, the objective function deforms to more non-convex ones. Hence, a proposed idea to alteration of p is choosing higher values of it in higher temperatures and decreasing p as temperature decreases.

Remark 3. The choice of C should *ideally* be such that the energy of points satisfying the cardinality constraint remains unchanged and the energy of all points violating the cardinality constraint is increased. However, since the penalty function is continuous this is not possible. We have to dynamically make choices of C through the algorithm such that it best serves this purpose.

Remark 4. The values of $f(x)$ and $C\left(\|x\|_p^p - K\right)^2$ need to be comparable in Taylor expansions around a given centre, in order for the D-Wave system to generate meaningful results. Hence, choice of C depends on considering the centre and the neighbourhood. It also needs to be small enough so that it does not amplify the error of Taylor approximations.

Example 10 (Application to portfolio optimization). Given n assets indexed by integers $\{1, \cdots, n\}$, we let r_i be a constant called the *expected return* of asset i. A symmetric positive semi-definite $n \times n$ matrix $C = (c_{ij})$ is also given and represents a covariance matrix. The portfolio optimization problem (Markowitz model) with cardinality constraint is

$$\min_{x \in \mathbb{R}^n} \; f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_i\, x_j \tag{6}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i = 1\,, \tag{7}$$

$$\sum_{i=1}^{n} x_i r_i = r\,, \tag{8}$$

$$0 \le x_i \le 1 \quad \forall i \in \{1, \cdots, n\}\,, \tag{9}$$

$$\#\,\text{supp}(x) \le K\,. \tag{10}$$

We have implemented an algorithm that uses the technique of §4 to solve this problem. The only feature of the algorithm specific to this problem is the choice of vectors $\{b_1, \cdots, b_m\}$ from which we create Ising models. In fact, we can always assume that $m = \lfloor n/3 \rfloor$. This means that if the largest complete graph that can be embedded on the chimera of a D-Wave system is of 33 variables, we are capable of solving this optimization problem in the presence of as many as 99 assets.

To do this we group assets in triples, and for each group we associate one direction vector b_i. In each triple we choose one leading asset that changes precisely according to the direction b_i. The increment and decrement of the other two are uniquely determined by the two linear constraints (5.2) and (5.3).

8 Experimental results

In appendices, we have benchmarked our method to solve the problem of portfolio optimization addressed in Chang et al. [2] and Woodside-Oriakhi et al. [13]. Namely, we look at the problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \; f(x) &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_i\, x_j \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = 1\,, \\ & \sum_{i=1}^{n} x_i r_i = r\,, \\ & x_i \in \{0\} \cup [\ell_i, u_i] \;\forall i \in \{1, \cdots, n\}\,, \\ & \#\,\mathrm{supp}(x) = K\,. \end{aligned}$$

We have used the same databases as the mentioned papers. Namely, for the five market indices Hang Seng (Hong Kong), DAX 100 (Germany), FTSE 100 (UK), S&P 100 (USA), and the Nikkei 225 (Japan), we used the datasets provided at `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/portinfo.html`. For S&P 500 (USA), and Russell 2000 (USA), we used `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/indtrackinfo.html`. We have set the cardinality constraint parameter to $K = 10$ and used lower and upper bounds $\ell_i = 0.01$ and $u_i = 1$ as in case of [2] and [13].

In our examples, the temperature schedule is the Boltzmann schedule (inverse of logarithm), and the selection scheme, is the choice of the best point out of all new points (hence always performing a jump, as if in very high temperature, but to the minimum-valued point in the neighbourhood). The norm parameter p explained above, has a geometric variation throughout the annealing process.

To test the precision of our results, we have used the same *gap measure* introduced by Chang et al. [2] and have reported our gaps in the appended table.

References

[1] C. J. P. Bélisle. Convergence theorems for a class of simulated annealing algorithms on $\mathbb{R}^d$. *Journal of Applied Probability*, pages 885–895, 1992.

[2] T.-J. Chang, N. Meade, J. E. Beasley, and Y. M. Sharaiha. Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research*, 27(13):1271–1302, 2000.

[3] Ambedkar Dukkipati, M. Narasimha Murty, and Shalabh Bhatnagar. Cauchy annealing schedule: An annealing schedule for boltzmann selection scheme in evolutionary algorithms. *CoRR*, cs.AI/0408055, 2004.

[4] Bruce Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2):311–329, May 1988.

[5] David E. Kaufman and Robert L. Smith. Direction choice for accelerated convergence in hit-and-run sampling. *Operations Research*, 46(1):84–95, 1998.

[6] R. Kindermann, J.L. Snell, and American Mathematical Society. *Markov Random Fields and Their Applications*. AMS books online. American Mathematical Society, 1980.

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[8] P. J. M. Laarhoven and E. H. L. Aarts, editors. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[9] M. Locatelli. Simulated annealing algorithms for continuous global optimization: Convergence conditions. *Journal of Optimization Theory and Applications*, 104(1):121–133, 01 2000. Copyright - Plenum Publishing Corporation 2000; Last updated - 2010-06-06.

[10] Shmuel Onn and Ishay Weissman. Generating uniform random vectors over a simplex with implications to the volume of a certain polytope and to multivariate extremes. *Annals of Operations Research*, 189(1):331–342, 2011.

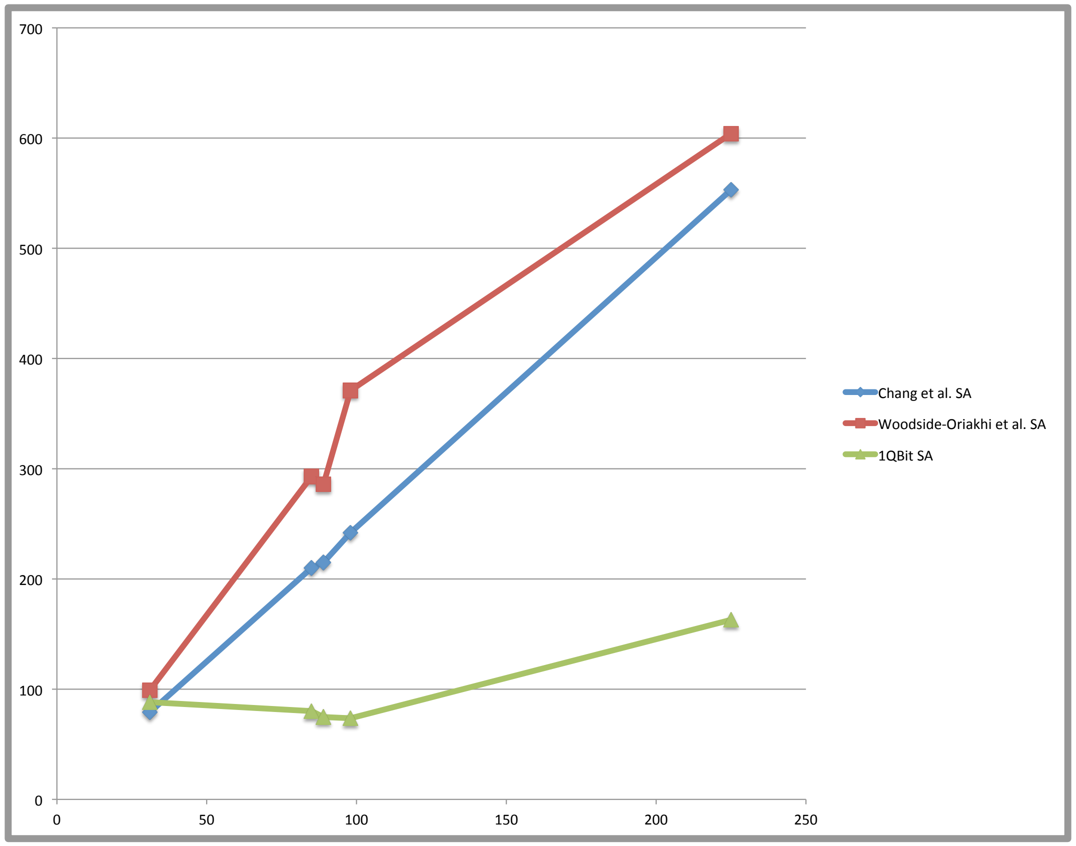
[11] H. Szu and R. Hartley. Fast simulated annealing. *Physics Letters A*, 122:157–162, June 1987.

[12] David Vanderbilt and Steven G Louie. A monte carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56(2):259–271, 1984.

[13] M. Woodside-Oriakhi, C. Lucas, and J.E. Beasley. Heuristic algorithms for the cardinality constrained efficient frontier. *European Journal of Operational Research*, 213(3):538–550, 2011.

©2016 1QB Information Technologies

	N	Gap measure and time	Chang et al. SA	Woodside-Oriakhi et al. SA	Mean of 1QBit
Hang Seng	31	Mean	0.99	1.06	0.05
		Median	1.21	0.54	0.00
		Min		0.03	0.00
		Max		4.64	1.11
		Time (s)	79	99	88
DAX 100	85	Mean	2.43	1.03	4.18
		Median	2.47	0.87	0.62
		Min		0.03	0.00
		Max		4.41	20.23
		Time (s)	210	293	80
FTSE 100	89	Mean	1.13	0.90	6.36
		Median	0.71	0.39	2.64
		Min		0.02	0.00
		Max		10.20	59.31
		Time (s)	215	286	75
S&P 100	98	Mean	2.70	3.10	9.72
		Median	1.13	2.11	8.61
		Min		0.87	0.00
		Max		8.67	32.19
		Time (s)	242	371	74
Nikkei 225	225	Mean	0.64	1.12	2.23
		Median	0.63	0.69	0.70
		Min		0.01	0.00
		Max		3.97	27.99
		Time (s)	553	604	163
			Silicon Graphics Indigo Station R4000, 100MHz, 48MB RAM	Intel Core2 PC 2.40 GHz, 3.24GB RAM	Intel Core2 Duo MAC, 2.26 GHz, 8GB RAM


Chang et al. SA
Woodside-Oriakhi et al. SA
1QBit SA