on Fundamentals of Electronics, Communications and Computer Sciences

DOI:10.1587/transfun.2020KEP0007

Publicized:2021/07/08

This article has been accepted and published on J-STAGE in advance of copyediting. Content is final as presented.

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3chome, Minato-ku, TOKYO, 105-0011 JAPAN

Analysis and Acceleration of the Quadratic Knapsack Problem on an Ising Machine

Matthieu PARIZY^{†a)} and Nozomu TOGAWA^{††b)}, *Members*

SUMMARY The binary quadratic knapsack problem (QKP) aims at optimizing a quadratic cost function within a single knapsack. Its applications and difficulty make it appealing for various industrial fields. In this paper we present an efficient strategy to solve the problem by modeling it as an Ising spin model using an Ising machine to search for its ground state which translates to the optimal solution of the problem. Secondly, in order to facilitate the search, we propose a novel technique to visualize the landscape of the search and demonstrate how difficult it is to solve QKP on an Ising machine. Finally, we propose two software solution improvement algorithms to efficiently solve QKP on an Ising machine.

key words: *Ising model, Ising machine, knapsack problem, visualization*

1. Introduction

1.1 The Binary Quadratic Knapsack Problem

The binary quadratic knapsack problem (QKP) is the problem of finding the combination items giving the highest profit within the capacity C of a knapsack for a given triangular profit matrix $P = \{p_{ij}\}$ of positive values. Let x_i be a binary variable showing whether the item i is chosen or not. If $x_i = 1$, the item i is chosen. p_{ij} is the profit when both item i and item j are chosen. p_{ii} is the profit when only item i is chosen. Each item i have a weight w_i :

$$\text{maximize } \sum_{i=1}^n \sum_{j=1}^n p_{ij} x_i x_j \quad (1)$$

$$\text{subject to } \sum_{i=1}^n w_i x_i \leq C \quad x_i \in \{0, 1\} \quad (2)$$

It is a generalization of the binary knapsack problem (where $p_{ij} = 0$ for $i \neq j$) as well as several other combinatorial problems which can also be expressed as a QKP such as the clique problem, and is considered a difficult NP-hard problem [1] [2].

Its known applications are VLSI design [3], compiler design [4], budgeting problem [5] and network flows [6].

QKP has been investigated for more than 40 years since its introduction by Gallo et al. [7]. Various approaches

ranging from exact algorithms, heuristics, meta-heuristics, linearization have been tried. The most efficient current approach to our knowledge comes from C. Patvardhan et al. using a quantum inspired evolutionary algorithm (QIEA) [8] and will thus be the base for comparing our results. Although the QKP was first formulated as an Ising model in [9] by Glover et al., to our knowledge, results of running this Ising model on an Ising machine have yet to be published.

1.2 Ising Machines

Ising machines have been gaining momentum over the course of the last 10 years with the advent of commercialization of quantum annealers [10] and other specialized architectures targeting the same problem. They take for input an Ising spin model and search for the lowest energy configuration of spins (which can have an "up" or "down" value, translatable to 1 or 0) i.e. the ground state. Numerous kinds of combinatorial problems can be mapped to an Ising model, such as QKP. Solving these mapped problems on Ising machines allows for certain categories of problem to be solved more efficiently than conventional methods. They represent an attempt to circumvent Moore's Law [11] [12].

QKP's quadratic objective function seemed like a good fit as it can be translated "as is" in an Ising model, although the capacity constraint had to be converted to an objective function using binary auxiliary variables. The broad applications of QKP, the fit to Ising model and the novelty of Ising machines is what motivated our research.

1.3 Our Proposal

In this paper we propose to analyze the results of running Ising models of QKP on an Ising machine on benchmark instances. Firstly, we run them using a naive formulation without any particular enhancements, secondly we analyze the bottlenecks of this first run, finally we propose software solution improvement algorithms to improve the results.

1.4 Contributions

Our main contributions are the following to improve convergence to optimal solution:

- Visualization techniques of Ising models which inspired the following contribution.
- Low α coefficient for the constraint part of the Hamiltonian coupled with a software solution improvement

[†]The author is with the company FUJITSU LABORATORIES LTD. Kawasaki, 211-8588 Japan.

^{††}The author is with the Department of Computer Science and Communications Engineering, Waseda University, Tokyo, 169-8555 Japan.

a) E-mail: parizy.matthieu@fujitsu.com

b) E-mail: ntogawa@waseda.jp

technique to "mend" non-feasible solutions and improve feasible ones into high quality feasible solutions.

Our benchmark used in this paper relies on instances from the work of Billionnet et al. [13]. To our knowledge, since the publication of [2], quantum inspired evolution algorithm approach from Patvardhan et Al. [8] is the approach yielding the best results both time wise and chance of solving an instance, it will thus be used for comparison with our results.

We do not claim superiority of Ising machines for QKP, but we want to share the various shortcomings we had and the techniques we came up with to overcome those and benchmark those techniques efficiency within the paradigm of finding the best solution using Ising models.

2. Simple QKP Trial on an Ising machine

In this section we present the results of the Ising models formulation found in [9] on instances taken from [13] using an Ising machine called a "Digital Annealer" (DA) [14],[15] as well as a "simulated annealing sampler"(SA) [16] for a baseline comparison.

2.1 Benchmark Instances Description

We used QKP instances from [13] which are all combinations of variable number $n = [100, 200, 300]$, profit matrix density $\Delta = [0.25, 0.5, 0.75, 1]$ randomly generated 10 times. For $n = 300$ only $\Delta = [0.25, 0.5]$ are present. Weight values have a normal distribution between 1 and 50. Profit values have a normal distribution between 1 and 100. Capacity is randomly distributed in $[50, \sum_i w_i]$.

2.2 QKP Ising Formulation

The first formulation of QKP as an Ising model or Hamiltonian was done by Glover et al. [9]. It is the following:

$$H = \sum_{i=1}^n \sum_{j=1}^n (-p_{ij} x_i x_j) + \alpha \left(\sum_{i=1}^n w_i x_i - C + y \right)^2 \quad (3)$$

where p_{ij} is the profit when both item i and item j are chosen, x_i is the binary variable representing if item i is chosen or not, w_i is the weight of item i , C is the weight capacity of the knapsack and y is the auxiliary integer variable for when the capacity is not fully used with $0 \leq y < C$. H is called Hamiltonian energy or energy for short.

The integer auxiliary variable y follows a one-hot encoding is expressed by binary variables y_i as follows:

$$y = \sum_{i=0}^m i y_i \text{ with } y \in \mathbb{Z} \text{ and } y_i \in \{0, 1\} \quad (4)$$

$$\text{subject to: } \left(\sum_{i=0}^m y_i - 1 \right)^2 = 0 \quad (5)$$

Auxiliary variables only need to compensate for values up to the maximum value minus one of an item in an instance of [13] as any values above would mean another item could be added in the knapsack as the max possible capacity of the knapsack in any instance is $\sum_i w_i$. Therefore $m = \max w_i - 1$. The reason for limiting the auxiliary variable range is to limit the number of iterations consumed flipping those variables.

We set the constraint coefficient α in Eqn. (3) as $strength \times n \times \Delta$ with $strength$ initial value set to 0.1, n being the problem size and Δ the profit matrix density. The larger and denser a problem is, the more adding an item to the knapsack can generate higher profit. Thus, α must be set to a higher value accordingly as Δ and n grow to prevent penalty violation.

If all the replicas (detailed will be explained in Section 2.3) of the DA converged to an infeasible solution, we judged that $strength$ was too low and increased it by 50%. We repeat this process until at least one feasible solution was reached.

2.3 Digital Annealer and its Hyper-Parameters

The DA operating principle is searching for the ground state of an Ising model using the basis of the Markov chain Monte Carlo (MCMC) method [15]. In the model used for the presented experiments, up to 8,192 variables can be defined with up to 67,108,864 couplers to define the weight binding the variables with a precision up to 64 bits (i.e. any variable x_i can be unconditionally connected to any variable x_j for a "full connectivity"). We choose the DA as its full connectivity makes it attractive for QKP with a high profit matrix density as they can be run without any transformation.

Another specificity of the DA is to have multiple processes, or replicas, doing the same MCMC search, at different temperatures, to speedup finding the ground state [15]. All the r replicas are given the same Ising model with their own different temperatures, evenly spread on a logarithmic scale between T_{max} and T_{min} , set by the user. Temperature wise adjacent replicas exchange their state, which is their lowest energy found solution, every "exchange rate" iterations. This parallel search is also known as parallel tempering [17]. Its main merit is to be less sensible to temperature scheduling in comparison to simulated annealing, as search will not halt on a local minimum if temperature is too low. Another DA specific mechanism is the "dynamic offset" [14]: every time a replica does not flip any binary variable in an iteration, DA applies an energy offset allowance, set by a corresponding hyper parameter called "offset_increase_rate", letting DA accept a binary flip causing an energy increase less or equal than the accumulated offset allowance, regardless of temperature. When a flip occurs, the offset allowance is reset to 0.

For our experiments we used parallel tempering with $r = 26$ replicas with $T_{max} = 9000$ and $T_{min} = 0$, without any temperature decay, with a replica temperature exchange rate of 100 iterations and an offset_increase_rate value of

Algorithm 1 SA Solution Mending Algorithm

```

1: procedure SOLUTIONMENDING
2: input:
3:   Initial SA solution:  $s = \{x_1, x_2, \dots, x_n\}$ 
4: output:
5:   Mended solution:  $s' = \{x_1', x_2', \dots, x_n'\}$ 
6: main:
7:    $s' = s$ ;
8:   for every  $x_k$  in  $s' \mid x_k = 1$  do
9:     Evaluate  $x_k$  removal cost loss;
10:  Accept the item removal giving the minimum cost loss;
11:  Repeat the above process until  $s'$  is feasible;
12:  return  $s'$ 

```

100.[†] Thus every time a replica does not flip any binary variable in an iteration, we apply an energy offset allowance of +100 and then reset when a flip occurs. All problems are ran over 750 million iteration which takes about 39 seconds with the above settings. Every result is evaluated over 20 different random seeds. Finally, the initial value of every binary variable is set to 0.

2.4 Simulated Annealing

To provide a baseline to compare with DA results, we chose the SA from D-Wave [16], which can perform simulated annealing on an Ising model. There are two reasons for this choice. First, as far as we know, no SA implementation specialized for the QKP have been proposed. The closest implementations we found were respectively for the quadratic multiple knapsack problem [19] and the generalized quadratic multiple knapsack [20] but they are not the same as our QKP. Secondly, the advantage of Ising model solvers is their ability to solve any combinatorial problem mapped to an Ising model, whereas specialized SA implementations are limited to the problem they are applied to. Thus, for a fair comparison, we decided to choose a software implementation of SA for Ising models.

We ran the SA with the same α , T_{max} and T_{min} as DA. We adjusted the SA number of iterations so that it would take 39 seconds, the same time as our DA experiments. This gave an SA number of iterations of 2,750,000. As for DA, every result is evaluated over 20 different random seeds and every binary variable initial value is set to 0. We also ran an another experiment with double the number of iterations for a total of 5,500,000 for reference.

Because some SA runs would lead to a constraint-violating solution, we applied the lightweight Algorithm 1 to mend these solutions to feasible solutions, whereas for DA, we keep as solution the best non constraint violating solution within all the replicas and our α policy ensures at least one replica reached a feasible solution.

2.5 First Benchmark Results

Tables 1, 2 and 3 show the number of times optimal answer

[†]Those parameters are the one which worked best for us over all instances after tuning them using *hyperopt* [18].

was reached over the 20 random seeds for each QKP instance of size 100, 200 and 300 respectively as "DA success%", "SA1 success%" and "SA2 success%", respectively for a 39s DA run, 39s SA run and 78s SA run, when trying to find the ground state of the Ising model. Likewise, "DA gap%", "SA1 gap%" and "SA2 gap%" are the mean gap achieved for the 20 seeds used. Gap is measured as described in Eqn. 6:

$$Gap = \frac{opt_val - obtainedValue}{opt_val} \times 100 \quad (6)$$

where *opt_val* is the known optimal profit value in every QKP instance from [8] and *obtainedValue* is the value reached by DA or SA. "Instance name" is the name of the QKP instance used. Δ is the profit matrix density expressed in %. The experiments on SA1 and SA2 were conducted on an Intel Core i7-9700K, 8 cores CPU with a base frequency of 3.6 GHz, using python 3.7.9 and dwave-neal 0.5.7.

We observed that in all problem instances SA could not reach optimal solution once, even for 78s runs. We also observed that in all instances, except r_100_75_1, r_200_100_1, r_200_100_8 and r_300_50_3, DA gap is much closer to optimal than SA1 and SA2. The average gap% over all instances for SA1 is 18.58%, for SA2 18.00% against 8.15% for DA. The average success% over all instances for SA1 and SA2 is none against 6.7% for DA. Whereas in [8] success% of 100% for all instances within less than 0.02 seconds is achieved with QIEA.

In Table 2 and Table 3, having larger problem instances only make things worse: optimal value solution found for only a single instance with DA.

3. Identifying Bottlenecks Using Visualization

Starting from the DA benchmark results from Section 2, we picked instance "r_100_25_10", with penalty Hamiltonian $\alpha = 5.5$, where optimal solution was difficult to reach for some random seeds and came up with the following algorithm for analyzing what makes reaching optimal solution difficult.

The key goal is to approximate the Hamiltonian energy landscape from an easy to reach sub-optimal solution to an optimal solution to be able to pinpoint what makes a problem hard to solve and eventually give us idea on how to tackle this difficulty. The procedure to approximate the landscape is as follows in Algorithm 2.

In Algorithm 2, a set $\{x_1, \dots, x_n\}$ of binary variables shows the sub-optimal solution for QKP and a set $\{y_1, \dots, y_n\}$ of binary variables shows the final optimal solution. The procedure can be described as:

1. Extract the non-common variables between the sub-optimal solution "start" and optimal solution "end".
2. Calculate and record which flip of non-common variable in start would cause the lowest energy increase.
3. Do the actual flip and repeat from 2. until start = end.

Thus $(x_a, x_b, \dots, x_k)$ is the order in which to

Table 1 100 variables problem results.

Instance name	Δ	opt_val	DA gap%	SA1 gap%	SA2 gap%	DA success%	SA1 success%	SA2 success%
r_100_25_1	25.0	18,558.0	0.6	12.1	12.3	5.0	0	0
r_100_25_10	25.0	24,930.0	0.0	8.7	8.4	30.0	0	0
r_100_25_2	25.0	56,525.0	2.9	12.3	12.5	0.0	0	0
r_100_25_3	25.0	3,752.0	0.0	32.0	31.0	100.0	0	0
r_100_25_4	25.0	50,382.0	4.5	14.2	14.6	0.0	0	0
r_100_25_5	25.0	61,494.0	1.7	6.7	6.2	0.0	0	0
r_100_25_6	25.0	36,360.0	0.7	9.5	8.6	0.0	0	0
r_100_25_7	25.0	14,657.0	0.4	10.4	11.3	0.0	0	0
r_100_25_8	25.0	20,452.0	0.4	12.9	11.3	20.0	0	0
r_100_25_9	25.0	35,438.0	2.6	12.0	11.3	0.0	0	0
r_100_50_1	50.0	83,742.0	1.8	10.2	9.9	0.0	0	0
r_100_50_10	50.0	88,634.0	1.4	9.3	8.6	0.0	0	0
r_100_50_2	50.0	104,856.0	2.7	14.2	13.0	0.0	0	0
r_100_50_3	50.0	34,006.0	0.0	12.6	13.7	55.0	0	0
r_100_50_4	50.0	105,996.0	0.8	6.0	5.5	0.0	0	0
r_100_50_5	50.0	56,464.0	8.3	29.9	30.0	0.0	0	0
r_100_50_6	50.0	16,083.0	0.0	13.5	13.1	100.0	0	0
r_100_50_7	50.0	52,819.0	3.4	15.1	14.3	0.0	0	0
r_100_50_8	50.0	54,246.0	2.9	14.8	14.1	0.0	0	0
r_100_50_9	50.0	68,974.0	3.2	13.5	12.1	0.0	0	0
r_100_75_1	75.0	189,137.0	4.2	3.7	3.5	0.0	0	0
r_100_75_10	75.0	143,740.0	1.1	7.4	6.7	0.0	0	0
r_100_75_2	75.0	95,074.0	2.5	12.3	12.0	0.0	0	0
r_100_75_3	75.0	62,098.0	2.2	10.6	12.1	0.0	0	0
r_100_75_4	75.0	72,245.0	1.0	9.3	8.8	0.0	0	0
r_100_75_5	75.0	27,616.0	0.2	14.9	17.6	10.0	0	0
r_100_75_6	75.0	145,273.0	4.7	16.8	17.3	0.0	0	0
r_100_75_7	75.0	110,979.0	6.2	21.4	19.7	0.0	0	0
r_100_75_8	75.0	19,570.0	8.0	43.1	46.4	0.0	0	0
r_100_75_9	75.0	104,341.0	7.7	24.8	25.8	0.0	0	0
r_100_100_1	100.0	81,978.0	8.4	28.5	27.8	0.0	0	0
r_100_100_10	100.0	193,262.0	1.0	7.0	6.6	0.0	0	0
r_100_100_2	100.0	190,424.0	1.0	5.8	6.0	0.0	0	0
r_100_100_3	100.0	225,434.0	0.2	4.0	3.7	0.0	0	0
r_100_100_5	100.0	230,076.0	0.0	3.0	2.9	30.0	0	0
r_100_100_6	100.0	74,358.0	11.3	36.5	33.2	0.0	0	0
r_100_100_7	100.0	10,330.0	0.0	28.6	26.0	100.0	0	0
r_100_100_8	100.0	62,582.0	10.5	35.5	34.7	0.0	0	0
r_100_100_9	100.0	232,754.0	5.0	6.6	6.7	0.0	0	0

Algorithm 2 Visualization

```

1: procedure ENERGY_PERSPECTIVE
2: input:
3:    $start \leftarrow (x_1, \dots, x_n)$ 
4:    $end \leftarrow (y_1, \dots, y_n)$ 
5: output:
6:    $(x_a, x_b, \dots, x_k) \mid start_{a,b,\dots,k} \oplus end_{a,b,\dots,k} = 1$ 
7:    $energy_a, energy_{a,b}, \dots, energy_{a,b,\dots,k}$ 
8: main:
9:    $PerspectiveOrder = \Phi$ ;
10:   $Energies = \Phi$ ;
11:  while  $start \neq end$  do
12:     $diff \leftarrow \{x_k \mid start_k \oplus end_k = 1\}$ 
13:    for all  $x_k \in diff$  do
14:      Try all  $x_k$  flips and record the index  $k$  in  $PerspectiveOrder$ 
      and  $energy_k$  in  $Energies$  of the one giving the largest energy decrease
      (or smallest energy increase).
15:    Do the actual flip of the best  $x_k$ 
16:  return  $PerspectiveOrder, Energies$ 

```

$energy_a, energy_{a,b}, \dots, energy_{a,b,\dots,k}$ are the respective energies of flipping x_a , x_a then x_b, x_a then $x_b \dots$ then x_k from start.

We then plot in order $energy_a, energy_{a,b}, \dots, energy_{a,b,\dots,k}$ on the y axis, and on the x axis the number of flips done from start. It gives us the perspective in Figure 1.

We annotated the variables which were switched at each move in order to go from our best non-optimal solution (START) to one optimal solution (END). We also annotated their energy (y axis value) for clarity.

The key takeaways from this perspective are:

1. START and END are only two variable swap away from each other: $x_{74} \leftrightarrow x_{98}$ and $x_{59} \leftrightarrow x_{85}$.
2. $x_{59} \leftrightarrow x_{85}$ causes the largest spike on the landscape. ($\Delta E \approx 7 \times 10^3$)

flip bits such that the Hamiltonian energy, given by equation (3), increase is the minimum possible and

Tackling both takeaways will be the focus of respectively Sections 4.1 and 4.3.

Table 2 200 variables problem results.

Instance name	Δ	opt_val	DA gap%	SA1 gap%	SA2 gap%	DA success%	SA1 success%	SA2 success%
r_200_25_1	25.0	204,441.0	2.4	6.7	6.6	0.0	0	0
r_200_25_10	25.0	48,459.0	9.5	28.8	27.2	0.0	0	0
r_200_25_2	25.0	239,573.0	0.4	2.9	2.6	0.0	0	0
r_200_25_4	25.0	222,361.0	4.9	11.0	10.8	0.0	0	0
r_200_25_5	25.0	187,324.0	3.3	7.8	7.8	0.0	0	0
r_200_25_6	25.0	80,351.0	7.3	17.7	15.8	0.0	0	0
r_200_25_7	25.0	59,036.0	10.7	25.4	25.3	0.0	0	0
r_200_25_8	25.0	149,433.0	4.4	11.1	10.2	0.0	0	0
r_200_25_9	25.0	49,366.0	8.2	23.8	24.9	0.0	0	0
r_200_50_1	50.0	372,097.0	4.5	8.9	8.5	0.0	0	0
r_200_50_10	50.0	284,751.0	8.6	14.7	15.6	0.0	0	0
r_200_50_2	50.0	211,130.0	7.4	15.7	14.6	0.0	0	0
r_200_50_3	50.0	227,185.0	18.1	31.3	28.4	0.0	0	0
r_200_50_4	50.0	228,572.0	18.9	31.3	31.8	0.0	0	0
r_200_50_5	50.0	479,651.0	0.3	2.0	2.2	0.0	0	0
r_200_50_6	50.0	426,777.0	1.6	4.6	4.5	0.0	0	0
r_200_50_7	50.0	220,890.0	11.4	21.3	19.1	0.0	0	0
r_200_50_8	50.0	317,952.0	5.9	10.5	9.7	0.0	0	0
r_200_50_9	50.0	104,936.0	11.2	25.5	23.0	0.0	0	0
r_200_75_1	75.0	442,894.0	16.5	22.0	22.5	0.0	0	0
r_200_75_10	75.0	142,694.0	8.6	21.3	19.1	0.0	0	0
r_200_75_2	75.0	286,643.0	15.1	24.1	21.9	0.0	0	0
r_200_75_3	75.0	61,924.0	4.0	26.0	23.2	0.0	0	0
r_200_75_4	75.0	128,351.0	18.8	36.6	35.8	0.0	0	0
r_200_75_5	75.0	137,885.0	10.2	26.8	24.8	0.0	0	0
r_200_75_6	75.0	229,631.0	21.2	33.5	32.3	0.0	0	0
r_200_75_7	75.0	269,887.0	9.9	20.9	19.0	0.0	0	0
r_200_75_8	75.0	600,858.0	11.1	13.8	13.3	0.0	0	0
r_200_75_9	75.0	516,771.0	14.2	20.4	18.8	0.0	0	0
r_200_100_1	100.0	937,149.0	7.2	6.1	5.3	0.0	0	0
r_200_100_10	100.0	378,442.0	10.0	15.8	14.5	0.0	0	0
r_200_100_2	100.0	303,058.0	14.7	23.1	22.3	0.0	0	0
r_200_100_3	100.0	29,367.0	0.0	21.3	18.9	100.0	0	0
r_200_100_4	100.0	100,838.0	8.3	27.1	26.8	0.0	0	0
r_200_100_5	100.0	786,635.0	7.7	9.8	9.7	0.0	0	0
r_200_100_6	100.0	41,171.0	11.3	31.9	29.9	0.0	0	0
r_200_100_7	100.0	701,094.0	6.3	9.5	9.0	0.0	0	0
r_200_100_8	100.0	782,443.0	14.2	14.1	14.1	0.0	0	0
r_200_100_9	100.0	628,992.0	6.3	9.3	9.0	0.0	0	0

Fig. 1 Energy landscape example: r_100_25_10 instance with $\alpha = 5.5$.

4. Algorithms for Solution Improvement

4.1 Solution Improvement Algorithm v1

This algorithm is a simplified version of "Procedure Improv-

eLocal(P)" described in [8] adapted for an Ising machine running parallel trials on different replicas such as we have here. For all replicas' solution obtained, we try all possible item swaps from inside and outside the knapsack, do the swap which generates the largest profit increase (while not violating the weight constraint), and iterate until no improvement can be done, resulting in a computing intensive process to improve current solutions.

We illustrate our solution improvement algorithm v1 (SIA v1) in Figure 2 using a non-quadratic knapsack problem to simplify explanations. In this example, "W" represents the weight of an item and "P" the profit of this item and we have already a solution obtained. We consider all possible swaps to improve the profit while staying under the capacity limitation which leads us to "Swap1", replacing item3 by item5 and then "Swap2" which replaces item2 with item6. The item coloring represents profit over weight metric, green means a substantial amount of profit is generated for a weight unit whereas red is the opposite.

Unlike "ImproveLocal(P)" from [8], we do not consider

Table 3 300 variables problem results.

Instance name	Δ	opt_val	DA gap%	SA1 gap%	SA2 gap%	DA success%	SA1 success%	SA2 success%
r_300_25_1	25.0	29,140.0	15.2	43.9	43.7	0.0	0	0
r_300_25_10	25.0	383,377.0	9.4	14.5	14.0	0.0	0	0
r_300_25_2	25.0	281,990.0	20.8	30.9	29.3	0.0	0	0
r_300_25_4	25.0	444,759.0	13.2	19.8	18.8	0.0	0	0
r_300_25_5	25.0	14,988.0	5.6	36.8	37.4	0.0	0	0
r_300_25_6	25.0	269,782.0	22.9	34.0	33.3	0.0	0	0
r_300_25_7	25.0	485,263.0	4.2	7.8	7.4	0.0	0	0
r_300_25_8	25.0	9,343.0	0.0	30.4	27.6	100.0	0	0
r_300_25_9	25.0	250,761.0	12.8	20.1	20.3	0.0	0	0
r_300_50_1	50.0	513,379.0	35.0	36.4	36.6	0.0	0	0
r_300_50_10	50.0	996,070.0	4.2	5.9	5.9	0.0	0	0
r_300_50_2	50.0	105,543.0	41.7	52.3	50.1	0.0	0	0
r_300_50_3	50.0	875,788.0	17.5	16.8	16.9	0.0	0	0
r_300_50_4	50.0	307,124.0	40.7	41.2	42.9	0.0	0	0
r_300_50_5	50.0	727,820.0	22.2	24.2	23.7	0.0	0	0
r_300_50_6	50.0	734,053.0	22.7	23.4	23.4	0.0	0	0
r_300_50_7	50.0	43,595.0	2.3	24.3	21.2	0.0	0	0
r_300_50_8	50.0	767,977.0	22.7	23.1	22.9	0.0	0	0
r_300_50_9	50.0	761,351.0	9.3	12.6	12.2	0.0	0	0

Fig. 2 SIA v1 example.

the action of adding without removing an item to improve a solution. Our reasoning is that in a real world scenario, it is unlikely that a replacement of an heavy item inside the knapsack by a light item outside the knapsack would bring a profit increase, as heavy items tend to bring more value for one unit of weight than lighter ones, and saves 1 loop of trials per solution to be improved. We note that considering item adding might help on randomly generated instances such as here. The pseudo-code is shown on Algorithm 3.

4.2 Solution Improvement Algorithm v1 Results

We made the following experiment to evaluate our proposed solution improvement algorithm (SIA v1). We used the exact same DA parameters as in Section 2. We allowed SIA v1 to do as many passes as it does for 40 seconds, which is the same timeout value as for the DA. The experiments were

Algorithm 3 Solution Improvement Algorithm

```

1: procedure SOLUTIONIMPROVEMENT
2: input:
3:   Initial solutions:  $S = \{s^1, s^2, \dots, s^r\}$ 
4: output:
5:   Improved solutions:  $S' = \{s^{1'}, s^{2'}, \dots, s^{r' \prime}\}$ 
6: main:
7:    $S' = S$ ;
8:   for every  $s^i$  in  $S'$  do
9:     for every pair  $(x_k^i, x_l^i)$  in  $s^i$  |  $\{x_k^i, x_l^i\} = (1, 0)$  do
10:      Evaluate  $(x_k^i, x_l^i)$  swap cost;
11:      Accept the possible swap giving the maximum cost increase
      and update the solution  $s^i$ ;
12:   Repeat the above process until no further improvement possible
      within timeout value;
13: return  $S'$ 

```

Table 4 success% comparison.

	DA	DA + SIA v1
n		
100	11.54	68.33
200	2.56	61.67
300	5.26	13.95
mean	6.70	55.00

conducted on an Intel Core i7-9700K, 8 cores CPU with a base frequency of 3.6 GHz, using python 3.7.9 and NumPy 1.17.3 [21]. We recorded success% as described in Section 2. For space issues purposes we aggregated the average success% of each instance sizes in Table 4. The "mean" row is the average success% calculated over all instances from Tables 1, 2 and 3 regardless of their size for both DA and DA+SIA v1. Note that we will later include the full breakdown for the solution improvement algorithm v2.

We observe a very sharp increase in % of instances which can be solved using the SIA v1, especially for smaller instances (100,200 items). We note that for the largest in-

Table 5 "r_100_25_10" instance with $\alpha = 5.5$ metrics comparison.

	$\sum_j p_{ij}$	w_i	RPD	$\sum_{j \neq i} Q_{i,j}$	$Q_{i,i}$
mean	1,222.4	26.1	99.0	198,331.5	-53,403.9
std	255.7	13.6	175.9	102,745.1	27,551.5
min	728.0	1.0	18.2	7,666.0	-101,500.0
Q1	1,051.2	15.0	31.9	114,570.0	-77,596.0
med	1,212.0	25.5	45.8	194,233.0	-52,389.5
Q3	1,368.2	38.0	87.9	288,496.0	-30,975.0
max	2,206.0	50.0	1,201.0	378,400.0	-2,079.0
x_{59}	1,668.0	36.0	46.3	273,456.0	-73,584.0
x_{85}	1,162.0	38.0	30.6	288,496.0	-77,596.0

mean, std, min, Q1, med, Q3 and max are respectively the average, the standard deviation, the minimum, the first quartile, the median, the third quartile and the maximum.

stances (300) the increase is much smaller. We consider that it is due to the timeout of 40 seconds being enough for smaller instances to converge to optimal solution in the majority of instances whereas it is not enough for most 300 items instances.

4.3 Solution Improvement Algorithm v2

As we saw in the perspective from Figure 1, changing the values of some variables while trying to leave a local optimum in search of a global one can create energy spikes difficult to overcome. We first try to understand what causes them by analyzing the problem's data and its formalization for the same r_100_25_10 instance with $\alpha = 5.5$. First, we define a few metrics for statistical comparison:

- Potential sum of profit: $\sum_j p_{ij}$ if all items j connected to item i were also chosen.
- Item i 's weight: w_i
- Relative profit density (RPD): $\sum_j p_{ij}/w_i$ also described in [8].
- Constraint Hamiltonian's $\sum_{j \neq i} Q_{i,j}$ where $Q_{i,j}$ is the penalty value (the second term in Eqn. (3)) added to the energy for item i if all items j connected to i were also chosen.
- Constraint Hamiltonian's $Q_{i,i}$: the value subtracted to the energy when item i is chosen.

By comparing the distributions of the metrics described above with the items 59 and 85 (source of the energy spike in Figure 1), we observe that although x_{85} is chosen in most DA solutions, it has a weaker RPD than x_{59} and also weights heavier. We also noticed that the constraint Hamiltonian values are one order of magnitude larger than profit values. This difference of magnitude comes from the fact that we need a high enough α , to prevent having the global ground state of the Ising model having constraint penalties, and we need a limited auxiliary variable range to limit the number of iterations spent on flipping auxiliary variable. This difference of magnitude produces an energy landscape which drives an Ising machine to fill the knapsack in priority and generating the highest profit second. The Hamiltonian thus

Fig. 3 SIA v2 example.

Algorithm 4 Filtering for removal of items

```

1: procedure FILTERING
2:   input:
3:     Profit:  $p_{ij}$  ( $1 \leq i \leq n, 1 \leq j \leq n$ )
4:     Weight:  $w_i$  ( $1 \leq i \leq n$ )
5:     filterLimit
6:   output:
7:     Indexes:  $I = \{i_1, i_2, \dots, i_{filterLimit}\}$ 
8:   main:
9:     for every item  $i$  do
10:      Calculate the relative profit density  $rp_d(i) = \sum_j p_{ij}/w_i$ ;
11:   Sort  $rp_d(i)$  and pick up 1st to filterLimit-th items into  $I$ ;
12:   return  $I$ 

```

gives priority to heavier items regardless of how much profit they bring, and generates a high penalty when trying to remove such item, which in turn makes escape from a local minimum difficult for an Ising machine.

It gave us the idea to use the RPD metric help escape from local minimum while preserving computing resources. The key idea is to give priority to the items with the lowest RPD for removal from the knapsack. How many of those worst items we consider for removal is set by a new parameter we introduce: "filterLimit".

We illustrate how the filtering works in Figure 3. Here with a filterLimit of 2 rather than considering all the swaps such as in Figure 2, we consider only swapping out the worst items RPD wise: item2 and item 3.

The above translates to the pseudo-code described in Algorithm 4 with Profit and Weight being the QKP input and filterLimit the parameter described above. For Figure 3 if we set $filterLimit = 2$, the algorithm would thus return $I = [3, 2]$ since the item 3 has the smallest RPD and the item 2 has the second smallest RPD.

This new filtering procedure is called to filter x_k on for loop of line 9 of Algorithm 3 to only those belonging to the returned I . Note that x_l from the same line is not filtered. If we take Figure 3 for example, the filtering is only applied to

Fig. 4 Success% vs SIA v2 filterLimit.

Fig. 5 Success% vs DA iterations (log scale).

the content inside the knapsack but not on the outside. We present its results in the following sub-section.

4.4 Solution Improvement Algorithm v2 Results

We used the environment described in Section 4.2 to experiment with different values for the newly introduced filterLimit parameter as well as the number of DA iterations. We observed its results in Figure 4. Best results are obtained with a filterLimit of 15. It represents an increase of 5% over the results from Section 4.2. There is also a significant variation of success% depending on the number of DA iterations.

Note that a high filterLimit does not allow to reach the results of SIA v1 which we could consider as "no filter limit". We assume it is due to the filtering being consuming due to the dot product and sorting involved, thus limiting the number of iterations which can be done within 40 seconds. We can also note that reducing the number of DA iterations does not degrade achieved success% much (going from 750 million iterations to 10 million only degrade success% of 7% at the best filterLimit value(15)).

We then focused on observing how success% evolves for a larger sample of DA iterations: [500k,...750M] and values for filterLimit: [7,10,15] as shown in Figure 5. We observed that although success% dramatically improves from 500k up to 200M iterations, it nearly plateaus (even on log scale) from 200M up to 750M with the best results being achieved for 500 million DA iterations.

We then focused on how success% evolves with the SIA v2 timeout value, for the sweet spot of 200M DA iterations, filterLimit:15 as shown in Figure 6. For small instances (100 items) even the minimum value of SIA v2 timeout we tried (5s) is enough to maximize success%. For average instances (200 items), 15 seconds is enough. For large instances (300 items), success% maxes out at 30 seconds.

Fig. 6 Success% vs SIA v2 timeout.

5. Results Comparison for Each Instance

Here we finally compare the success% achieved for: DA only; SIA v1 timeout 40s; and SIA v2 filterLimit:15, DA iterations:750M, timeout:40s.

From Table 6 we observe that SIAv1 and v2 improve the success% overall but for instances r_100_25_1, r_100_25_10, r_100_25_3, r_100_25_8, r_100_50_3 and r_100_100_7 DA alone performs better. We consider that it might be due to either the fact that SIA does not consider simply adding items in the knapsack and/or that the penalty Hamiltonian α being too weak makes all replicas converge to the same penalty violating state and from that state, using SIA we can only reach a feasible local minimum.

In future works, we will consider revising SIA to consider item adding and our α increasing policy. We will also keep the best DA solution for fallback in case SIA performs worse than SIA so that SIA results will be in the worst situation equal to DA results.

On Table 7 and Table 8, we observe a similar trend to Table 6: the only instance which used to be solved consis-

Table 6 100 variables problem results comparison.

Instance name	Δ	DA only	DA + SIAv1	DA + SIAv2
r_100_25_1	25	5.0	0.0	0.0
r_100_25_10	25	30.0	0.0	0.0
r_100_25_2	25	0.0	100.0	100.0
r_100_25_3	25	100.0	0.0	0.0
r_100_25_4	25	0.0	100.0	100.0
r_100_25_5	25	0.0	100.0	100.0
r_100_25_6	25	0.0	0.0	0.0
r_100_25_7	25	0.0	100.0	100.0
r_100_25_8	25	20.0	0.0	0.0
r_100_25_9	25	0.0	100.0	100.0
r_100_50_1	50	0.0	100.0	100.0
r_100_50_10	50	0.0	10.0	15.0
r_100_50_2	50	0.0	100.0	100.0
r_100_50_3	50	55.0	0.0	0.0
r_100_50_4	50	0.0	100.0	100.0
r_100_50_5	50	0.0	100.0	100.0
r_100_50_6	50	100.0	100.0	100.0
r_100_50_7	50	0.0	100.0	100.0
r_100_50_8	50	0.0	100.0	100.0
r_100_50_9	50	0.0	100.0	100.0
r_100_75_1	75	0.0	100.0	100.0
r_100_75_10	75	0.0	100.0	100.0
r_100_75_2	75	0.0	45.0	50.0
r_100_75_3	75	0.0	100.0	100.0
r_100_75_4	75	0.0	80.0	90.0
r_100_75_5	75	10.0	100.0	100.0
r_100_75_6	75	0.0	100.0	100.0
r_100_75_7	75	0.0	100.0	100.0
r_100_75_8	75	0.0	100.0	100.0
r_100_75_9	75	0.0	30.0	30.0
r_100_100_1	100	0.0	0.0	0.0
r_100_100_10	100	0.0	0.0	0.0
r_100_100_2	100	0.0	100.0	100.0
r_100_100_3	100	0.0	100.0	100.0
r_100_100_5	100	30.0	100.0	100.0
r_100_100_6	100	0.0	100.0	100.0
r_100_100_7	100	100.0	0.0	0.0
r_100_100_8	100	0.0	100.0	100.0
r_100_100_9	100	0.0	0.0	0.0

Table 7 200 variables problem results comparison.

Instance name	Δ	DA only	DA + SIAv1	DA + SIAv2
r_200_25_1	25	0.0	100.0	100.0
r_200_25_10	25	0.0	100.0	100.0
r_200_25_2	25	0.0	100.0	100.0
r_200_25_4	25	0.0	100.0	100.0
r_200_25_5	25	0.0	60.0	75.0
r_200_25_6	25	0.0	85.0	70.0
r_200_25_7	25	0.0	100.0	100.0
r_200_25_8	25	0.0	100.0	100.0
r_200_25_9	25	0.0	100.0	100.0
r_200_50_1	50	0.0	100.0	100.0
r_200_50_10	50	0.0	50.0	30.0
r_200_50_2	50	0.0	10.0	5.0
r_200_50_3	50	0.0	100.0	100.0
r_200_50_4	50	0.0	100.0	100.0
r_200_50_5	50	0.0	100.0	100.0
r_200_50_6	50	0.0	85.0	85.0
r_200_50_7	50	0.0	40.0	20.0
r_200_50_8	50	0.0	30.0	100.0
r_200_50_9	50	0.0	100.0	100.0
r_200_75_1	75	0.0	0.0	0.0
r_200_75_10	75	0.0	0.0	0.0
r_200_75_2	75	0.0	40.0	70.0
r_200_75_3	75	0.0	0.0	0.0
r_200_75_4	75	0.0	100.0	100.0
r_200_75_5	75	0.0	0.0	0.0
r_200_75_6	75	0.0	100.0	100.0
r_200_75_7	75	0.0	100.0	100.0
r_200_75_8	75	0.0	10.0	55.0
r_200_75_9	75	0.0	65.0	100.0
r_200_100_1	100	0.0	60.0	15.0
r_200_100_10	100	0.0	100.0	100.0
r_200_100_2	100	0.0	0.0	0.0
r_200_100_3	100	100.0	0.0	0.0
r_200_100_4	100	0.0	100.0	100.0
r_200_100_5	100	0.0	5.0	30.0
r_200_100_6	100	0.0	100.0	100.0
r_200_100_7	100	0.0	10.0	30.0
r_200_100_8	100	0.0	50.0	90.0
r_200_100_9	100	0.0	5.0	15.0

tently by DA is not solved by either SIA. Overall success% is improved by SIA v1 and v2, v2 giving the better results. Since overall SA performance is poor compared to DA as seen in Section 2.5 and Solution Improvement Algorithm (SIA) solution quality is assumed to be correlated to input solution quality, we did not run any SA+SIA experiments. Our proposed DA+SIA method is thus superior to SA1 and SA2 as well as SA1+SIA and SA2+SIA.

6. Conclusions

In this paper, we proposed a way to visualize the energy landscape between two states. This visualization method is based on the Ising model itself and solutions found which represent states of the Ising model and thus can be applied when using other Ising machines. It helped us conceive our other proposal: a software solution improvement technique coupled with low α value which can compensate for the problem of a low auxiliary variable range giving too much priority to filling the knapsack over finding the solu-

tion which generates the most profit. Our software solution improvement algorithms make handling heavy items more efficient by considering swapping out the one with the lowest RPD them in priority with other items. They operate on the original QKP formulation, before being mapped to its corresponding Ising model, and thus can operate on solutions found by other methods than Ising machines.

Our experiments showed promising results summarized in Table 9. In the "mean" row, we compared the average success% over all instances from Tables 1, 2 and 3 for DA, DA+SIA v1, DA+SIAv2, respectively. We went from an overall success% of 6.7% to a success% of 60.7%. We believe that our α adjustment policy is reasonable yet coarse grained and thus can potentially be improved. We also believe considering adding items without swapping could improve results. In future works we will also continue to develop novel ways to analyze Ising models and their behavior on Ising machines as well as test our current algorithm on other kind of problems to evaluate how much potential they have for a more general application, not limited to QKP.

Table 8 300 variables problem results comparison.

Instance name	Δ	DA only	DA + SIAv1	DA + SIAv2
r_300_25_1	25	0.0	100.0	100.0
r_300_25_10	25	0.0	0.0	35.0
r_300_25_2	25	0.0	0.0	55.0
r_300_25_4	25	0.0	0.0	20.0
r_300_25_5	25	0.0	0.0	0.0
r_300_25_6	25	0.0	5.0	5.0
r_300_25_7	25	0.0	0.0	60.0
r_300_25_8	25	100.0	0.0	0.0
r_300_25_9	25	0.0	5.0	20.0
r_300_50_1	50	0.0	0.0	100.0
r_300_50_10	50	0.0	0.0	30.0
r_300_50_2	50	0.0	100.0	100.0
r_300_50_3	50	0.0	0.0	0.0
r_300_50_4	50	0.0	55.0	70.0
r_300_50_5	50	0.0	0.0	20.0
r_300_50_6	50	0.0	0.0	0.0
r_300_50_7	50	0.0	0.0	0.0
r_300_50_8	50	0.0	0.0	0.0
r_300_50_9	50	0.0	0.0	0.0

Table 9 Result Summary.

	DA	DA + SIA v1	DA + SIA v2
size			
100	11.5	68.3	68.8
200	2.6	61.7	66.4
300	5.3	13.9	32.4
mean	6.7	55.0	60.7

References

- [1] A. Caprara, D. Pisinger, and P. Toth, "Exact solution of the quadratic knapsack problem," *INFORMS Journal on Computing*, vol.11, no.2, pp.125–137, 1999.
- [2] D. Pisinger, "The quadratic knapsack problem—a survey," *Discrete Applied Mathematics*, vol.155, no.5, pp.623–648, 2007.
- [3] C.E. Ferreira, A. Martin, C.C. de Souza, R. Weismantel, and L.A. Wolsey, "Formulations and valid inequalities for the node capacitated graph partitioning problem," *Mathematical Programming*, vol.74, no.3, pp.247–266, 1996.
- [4] E.L. Johnson, A. Mehrotra, and G.L. Nemhauser, "Min-cut clustering," *Mathematical programming*, vol.62, no.1-3, pp.133–151, 1993.
- [5] D. Laughhunn, "Quadratic binary programming with application to capital-budgeting problems," *Operations research*, vol.18, no.3, pp.454–461, 1970.
- [6] J.M. Rhys, "A selection problem of shared fixed costs and network flows," *Management Science*, vol.17, no.3, pp.200–207, 1970.
- [7] G. Gallo, P.L. Hammer, and B. Simeone, *Quadratic knapsack problems*, pp.132–149, Springer Berlin Heidelberg, Berlin, Heidelberg, 1980.
- [8] C. Patvardhan, S. Bansal, and A. Srivastav, "Solving the 0–1 quadratic knapsack problem with a competitive quantum inspired evolutionary algorithm," *Journal of Computational and Applied Mathematics*, vol.285, pp.86–99, 2015.
- [9] F. Glover, G. Kochenberger, and B. Alidaee, "Solving quadratic knapsack problems by reformulation and tabu search: Single constraint case," *Combinatorial and Global Optimization*, vol.14, 04 2002.
- [10] M.W. Johnson, M.H.S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A.J. Berkley, J. Johansson, P. Bunyk, E.M. Chapple, C. Enderud, J.P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M.C. Thom, E. Tolkacheva, C.J.S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, "Quantum annealing with manufactured spins," *Nature*, vol.473, no.7346, pp.194–198, May 2011.
- [11] M. Babaeian, D.T. Nguyen, V. Demir, M. Akbulut, P.A. Blanche, Y. Kaneda, S. Guha, M.A. Neifeld, and N. Peyghambarian, "A single shot coherent ising machine based on a network of injection-locked multicore fiber lasers," *Nature Communications*, vol.10, no.1, p.3516, Aug 2019.
- [12] R. Hamerly, T. Inagaki, P.L. McMahon, D. Venturelli, A. Marandi, T. Onodera, E. Ng, C. Langrock, K. Inaba, T. Honjo, K. Enbutsu, T. Umeki, R. Kasahara, S. Utsunomiya, S. Kako, K.i. Kawarabayashi, R.L. Byer, M.M. Fejer, H. Mabuchi, D. Englund, E. Rieffel, H. Takesue, and Y. Yamamoto, "Experimental investigation of performance differences between coherent ising machines and a quantum annealer," *Science Advances*, vol.5, no.5, 2019.
- [13] A. Billionnet and F. Calmels, "Linear programming for the 0–1 quadratic knapsack problem," *European Journal of Operational Research*, vol.92, no.2, pp.310–325, 1996.
- [14] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H.G. Katzgraber, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Frontiers in Physics*, vol.7, p.48, 2019.
- [15] S. Matsubara, M. Takatsu, T. Miyazawa, T. Shibasaki, Y. Watanabe, K. Takemoto, and H. Tamura, "Digital annealer for high-speed solving of combinatorial optimization problems and its applications," 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), pp.667–672, 2020.
- [16] "D-wave simulated annealing sampler." <https://docs.ocean.dwavesys.com/projects/neal>. Accessed: 2021-03-10.
- [17] K. Hukushima and K. Nemoto, "Exchange monte carlo method and application to spin glass simulations," *Journal of the Physical Society of Japan*, vol.65, no.6, p.1604–1608, Jun 1996.
- [18] J. Bergstra, D. Yamins, and D.D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13*, p.I-115–I-123, JMLR.org, 2013.
- [19] Y. Chen and J.K. Hao, "Iterated responsive threshold search for the quadratic multiple knapsack problem," *Annals of Operations Research*, vol.226, no.1, pp.101–131, Mar 2015.
- [20] Y. Chen and J. Hao, "Memetic search for the generalized quadratic multiple knapsack problem," *IEEE Transactions on Evolutionary Computation*, vol.20, no.6, pp.908–923, 2016.
- [21] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del R'io, M. Wiebe, P. Peterson, P. G'érard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T.E. Oliphant, "Array programming with NumPy," *Nature*, vol.585, no.7825, pp.357–362, Sept. 2020.

Matthieu Parizy received his B.Eng and M.Eng from ESIEE Paris in 2006 and 2008 respectively, all in computer science. He then joined the company FUJITSU LABORATORIES LTD where he is currently working while being enrolled in a PhD course from Waseda University as of 2020. His research interests are combinatorial optimization, machine learning and VLSI design.

Nozomu Togawa received his B.Eng., M.Eng., and Dr.Eng. degrees from Waseda University in 1992, 1994, and 1997, respectively, all in electrical engineering. He is presently a Professor in the Department of Computer Science and Communications Engineering, Waseda University. His research interests are VLSI design, graph theory, and computational geometry. He is a member of IEEE, ACM, and IPSJ.