Fast Embedding of Constrained Satisfaction Problem to Quantum Annealer with Minimizing Chain Length

Juexiao Su[1] and Lei He[1]
[1]EE Dept., Univeristy of California, Los Angeles, CA, USA
sujuexiao@g.ucla.edu, lhe@ee.ucla.edu

ABSTRACT

Recent research has demonstrated promising results in solving constrained satisfaction problem (CSP) using D-Wave quantum annealer. However, the embedding of the CSP suffers drawbacks such as long embedding time in addition to poor quality due to long chains that reduce the ground state probability. To address those issues, we propose an effective embedding technique that reduces the embedding time and minimizes the chain length. We compared to the most recent method published in DAC 2016. Experiments using existing D-Wave 2X quantum annealer show that the proposed embedding technique increases the ground state probability by 29% on average. Furthermore, to demonstrate the efficiency, we embedded large problems onto a predicted C100 D-Wave Chimera architecture. Experimental results show that our approach reduces the run-time by 3.4x on average with reduced longest chain length.

1. INTRODUCTION

With the emergence of D-Wave quantum annealer, many researches have proposed techniques to solve various optimization problems using the D-Wave, towards achieving quantum supremacy. The D-Wave quantum annealer operates itself by finding the ground states that minimize the system Hamiltonian, in which the optimization problem is elaborately encoded. It comprises a superconducting system whose Hamiltonian can be described by the Ising model [1][2].

$$E(s) = \sum_{1 \le i \le N} h_i s_i + \sum_{1 \le i \le j \le N} J_{ij} s_i s_j \quad (1)$$

$$| h_i | \le 2, | J_{ij} | \le 1, and\, s_i \in \{+1, -1\} \quad (2)$$

As shown in Eq (1), s_i is the state of a single qubit, h_i is the qubit bias and J_{ij} is the interaction weight that acts on two qubits. Slightly different from Ising model, quadratic unconstrained binary optimization (QUBO) defines its variable x_i over $\{0, 1\}$. However, the two problems are equivalent because s_i and x_i are linearly inter-changeable. Both h_i and J_{ij} are programmable so that different QUBO can be implemented. It is immediately apparent that the quantum annealer is capable of solving any problems as long as the problem can be represented in the form of QUBO. However, in practice, not every QUBO problem can be directly solved by D-Wave annealer due to the limited connectivity in the physical architecture. Therefore, embedding the mapped QUBO problem to the annealer's architecture becomes a necessary step in using D-Wave quantum annealer.

The scalability of embedding techniques is of great importance, as the capacity of the D-Wave quantum annealer exponentially increases. Many researches [3][4][5][6] have studied a wide range of applications that can be mapped to QUBO including machine learning, protein folding, satisfiability problem and trading strategy. However, only a few researches discussed embedding techniques and those techniques are not even scalable to the current generation of D-Wave annealer when embedding certain types of QUBO problems [7][8][9]. It is a desire to have an efficient embedding technique that not only manages to handle the problem with the size of the current architecture but is also scalable to the future devices with much larger capacity.

Chain has been introduced to solve embedding problem [4][5][7], which ensures the multiple qubits act as a single one during quantum annealing. It helps to reduce the degree of the vertex in the original QUBO, so that the QUBO can be embedded into the architecture with low connectivity. Minimizing chain length is also an important objective in developing embedding techniques, as the size of a chain significantly impacts the performance of the quantum annealer.

In this work, we focus on the embedding of constraint satisfaction problem which is defined as finding the state of variables that satisfy a set of constraints or limitations. Both [5][10] proposed scalable techniques that models the embedding as a place-and-route problem in electronic design automation. We summarized the work flow for CSP as three major stages: (1) local embedding, (2) constraint placement, (3) chain routing.

Local embedding involves embedding each constraint into a sub-graph of the architecture. [11] proposed an effective and optimal approach to embed each individual constraint to the quantum annealer by leveraging the satisfiability module theory. Alternatively, [5] proposed an approach that decomposes a large constraint into basic logic operations so that even a complex constraint can be locally embedded.

During placement, constraints are spread out while the

DAC '17 June 18-22, 2017, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/3061639.3062246

ones that share the same variables are placed close to each other. The quality of the placement substantially affects the subsequent routing quality and efficiency. [5] proposed a simulated annealing based placement algorithm, whereas [10] proposed a combined place-and-route algorithm. Both algorithms managed to minimize the total chain length, although both of them are not aimed to reduce the number and the length of long chains.

Next, qubits that shares the same variables are connected by chains. [5] also proposed a scalable routing algorithm by using negotiation strategy to route all the constraints.

In this paper, we focus on improving the areas that have been overlooked by past researches. Specifically, we develop data structures and algorithms to optimize the chain length. The distinguished features in our study are summarized as follows.

- An incremental placement technique that employs winner tree and look up table data structure to speed up the cost calculation.
- An A* routing engine to accelerate the routing speed, based on the repeated pattern in the D-Wave topology.
- A new placement cost function that shortens the longest chain.
- A novel dynamic criticality technique to optimize the chain length and improve the routing convergence.
- A rip-up and reroute stage to further optimize the chain length.

To validate the effectiveness, we conducted two experiments. First, we embedded CSPs onto an hypothetical C100 D-Wave architecture which is over a hundred times larger than the latest available D-Wave annealer, experimental result shows that our approach achieves 3.4X speed-up with reduced longest chain size. In the second experiment, we embedded CSPs onto real D-Wave 2X quantum annealer, result suggests that our approach increases the ground state probability by 29% on average.

The rest of the paper is organized as follows: Section 2, background. Section 3, technical details of the proposed approach. Section 4, experimental results and section 5, conclusion.

2. PRELIMINARIES

2.1 Mapping CSP to QUBO

In this section we discuss the mapping of Constraint Satisfaction Problem (CSP) to QUBO problem. Practically, we confine our discussion within the scope of CSP with finite binary variables observing that the binary variable can be extended to other types with proper encoding. Here, the CSP is stated as, a set of binary variables $s_i \in \{-1, +1\}$ and a set of constraints, each of which contains a non-empty set C_i of variable assignment. The objective is to find an assignment $\boldsymbol{s} \in \bigcap C_i$.

[11] proposed following method to map CSP to QUBO. First, a penalty function is constructed for each constraint:

$$Penalty_{C_j}(\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} k & \text{if } \boldsymbol{s} \in C_i \\ \geq k + g & \text{if } \boldsymbol{s} \notin C_i \text{ and } g > 0 \end{cases} \quad (3)$$

where $\boldsymbol{s}$ is an assignment, $\boldsymbol{a}$ represents the auxiliary variables which essentially are the don't-care variables in the corresponding constraint. Secondly, as a variable may be shared by many different constraints, a chain is used to add penalty in the QUBO if the qubits that represent the same variable act differently in different constraints. Therefore, the QUBO is minimized if and only if an assignment satisfies all constraints.

Intuitively, a QUBO problem can be represented by a graph where each vertex represents a variable, and weights, as the h_i and J_{ij} in the Eq (1), are assigned to each vertex and edge respectively. Figure 1 shows a toy example of a CSP mapping with two constraints, $s_1 \oplus s_2 = s_3$ and $s_1 \oplus s_4 = s_5$.

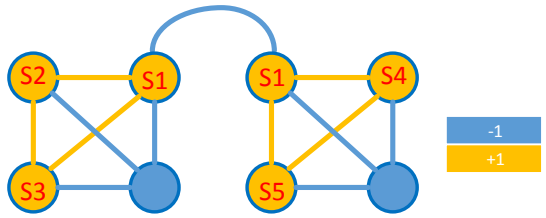

Figure 1: Toy example with two constraints. Vertexes are the variables. Different colors indicate the weight that is assigned to the vertexes or the edges

2.2 Challenges in Embedding

D-Wave annealer has a low connectivity. Its architecture is realized by arranging qubits according to the Chimera graph topology. D-Wave 2x comprises 12 by 12 cells. Each of which consists a $K_{4,4}$ bipartite graph, as shown in Figure 2. It is also noticeable that several vertexes and edges are inactive due to manufacture defects. As a result, each individual qubit is only able to interact with limited number of qubits in the same and adjacent cells.

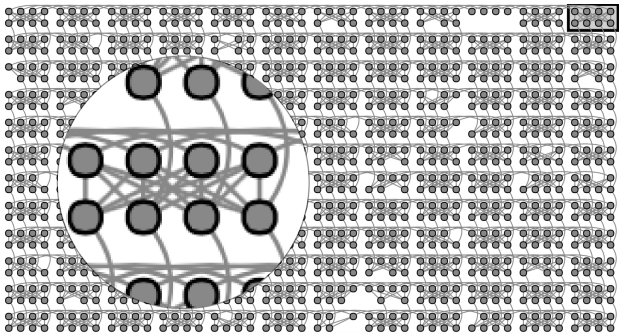

Figure 2: The D-Wave 2X Connectivity Graph

Moreover, the performance of quantum annealer is affected by the embedding result. D-Wave 2X annealer performs computation by quantum annealing [2], which is a non-deterministic procedure. Since it is impossible to guarantee that the annealer would find the ground state in each run, a straight forward strategy is to fire multiple runs and select the best from a set of candidate solution. Therefore, to assess the performance of quantum annealer, one of the most important standard is the probability to observe the ground state. The ground state probability depends on many factors, among which the longest chain size in the embedding

result is a major one. It is very hard to formulate the kinks in long chains, as their dynamics may slow down and impede the development of alignment during the transition between the initial state and the final state[12].

In short summary, the embedding technique should be able to efficiently handle the low connectivity architecture, at the same time, minimizing the longest chain length.

3. EMBEDDING FLOW

In the previous studies, [10] proposed an iterative combined place-and-route flow whereas [5] proposed a waterfall flow that separates the placement and routing. In this paper, we decide to use the waterfall flow for the following reasons.

- The waterfall flow clearly segregates the duties of placement and routing, hence the quality of placement and routing can be accurately measured. Furthermore, the result of each step also helps to fine tune the algorithm.
- The combined flow is likely to introduce instability that leads to unpredictable result. This is because the placement affects the routing, and in turns, the routing also affects the quality of placement, leading to a chicken-and-the-egg problem.
- The waterfall flow supports the integration of the path finder [13], which is a very efficient algorithm in solving the field programmable gate array (FPGA) routing problem. It is reasonable to believe that it is also very efficient in routing the chains, given that the routing graph in D-Wave is much smaller than that of FPGA.

In the remaining of this section, we discuss details of the proposed innovations in the placement and routing.

3.1 Constraint Placement

The quality of the placement greatly affects the subsequent routing quality. Based on our analysis in the previous sections, there are three objectives in the placement. (1) minimizing the total wire length, (2) minimizing those long chains that affects the ground state probability, (3) leaving enough space for routing.

Simulated annealing based placement has been proven to be very effective in achieving multiple optimization objective, so we adopt simulated annealing in our embedding flow. Algorithm 1 shows a general simulated annealing placement algorithm. In comparison with previous researches, our innovations in placement include a comprehensive placement cost function to achieve three optimization objectives, along with it, a delicate date structure to improve the computational efficiency.

3.1.1 Placement Cost Function

The quality of placement result depends heavily on the cost function. We propose the following cost function to factor in the chain length minimization, as shown in Eq (4).

$$Cost = \sum_{i \in Chains} q(i) \cdot [\frac{bb_x(i)}{(S_h(i))^m} + \frac{bb_y(i)}{(S_v(i))^m}] \cdot Crit(i) \quad (4)$$

where $Crit(i) = \sqrt{Chain_i / Chain_{longest}}$, indicating the criticality of each chain. The longer the chain, the higher its criticality. We take the square root of the length ratio to emphasize the importance of those chains that close to the longest chain, as the ratio is always smaller than one. $S_{v(h)}(i)$ is the routing supply in vertical(horizontal) direction, and $q(i)$ is the coefficient for multi-fanout chains.

Algorithm 1 Simulated Annealing Placement

```
M = move limit
T = initial temperature
for i ←0 to M do
    Generate move
    Evaluate Δcost
    if Δcost > 0 then
        Accept move
    else
        Accept with probability P = e^(−kΔcost/T)
    end if
    update T
end for
```

In D-Wave embedding flow, the routability is the key to the success of the subsequent routing stages. Unlike placement in circuit design, the routing resources per constraint are very limited, and the situation could be even worse if multiple constraints are placed in the same area as the constraint itself will consume some of routing resource. In the proposed cost function, we model the routing resource as the number of vertexes and edges that are available within the chain bounding box. m is a positive number to adjust the influence of the routing supply in the cost function and it can be fine tuned by experiment.

3.1.2 Incremental Cost Update

Apparently, an efficient way to compute the cost of the placement should considerably reduce the run-time in the placement. Based on the incremental bounding box update proposed by[14], we extend the incremental computation to routing supply and chain criticality.

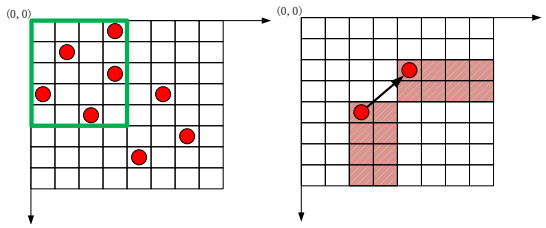

Figure 3: Routing Supply Update

We use a two-dimensional array to store the available routing resource with given placement. It is straight forward to use cells in the chimera graph to formulate the bins in the grid graph. We store the number of available routing resources in the top left area of $\{(0,0),(i,j)\}$ in $S(i,j)$, so that we can calculate the available routing resources in any bounding box that expands its area from (x_a, y_a) to (x_b, y_b) using the Eq (5). To incrementally update the available routing resource, we only need to update the $S(i,j)$ in the shading area, after the placement move is generated, as shown in Figure 3.

$$S = S(x_b, y_b) - S(x_a, y_b) - S(x_b, y_a) + S(x_a, y_b) \quad (5)$$

For the chain criticality, winner tree is used to support incremental update as shown in Figure 4. The bottom layer

stores the length of each chain, while each upper layer stores the winners from the lower layer. By doing this, we can update the winner caused by a single-chain-length change with complexity of $logN$, where N is the number of chains.

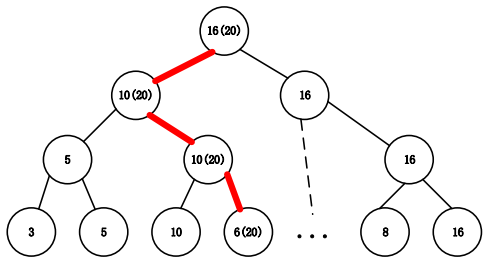

Figure 4: Winner Tree Chain Size Update

3.2 Chain Routing

[5] proposed a chain routing algorithm based on pathfinder. As an extension of their work, we proposed innovative methods that improve the run-time and minimize the chain length. In the remaining of this section, we first discuss the run-time speed-up techniques, and then present a rip-up and reroute algorithm that aggressively minimizes the longest chain length.

3.2.1 Routing Speed-up

We use A* to accelerate the shortest path algorithm, which employs a heuristic cost to guide the order of vertex wave expansion in Dijkstra shortest path algorithm. Here we use the shortest path length as the A* heuristic cost, as it can be conveniently calculated based on the periodic pattern in the D-Wave Chimera graph. Eq (6) shows the shortest path length calculation of any given vertex pair. Figure 5 shows an example of the shortest path with given source and destination vertex pair.

$$ShortestPath(v_1, v_2) = |v_{1x} - v_{2x}| + |v_{1y} - v_{2y}| + k \quad (6)$$

where x,y indicate the cell coordinates and k is used to add adjustment length if the v_1, v_2 are not sitting in the counterpart location of two different cells.

3.2.2 Dynamic Criticality

We propose the dynamic criticality D_{cr}-based negotiation routing. Eq (7)-(9) show the cost functions for each vertex. The terminology is similar to the pathfinder: C_b is the base cost, C_s is the congestion cost or share cost, and C_h is the history cost and P_i is the penalty factor that increases with the routing iterations. Unlike static criticality in pathfinder, we propose dynamic criticality to reduce the chain length and to accelerate routing convergence.

$$C_v = Dcr * C_b + (1 - Dcr) * (C_b + C_s * P_i + C_h) \quad (7)$$

$$L_{pred} = (L_c + L_d) \quad (8)$$

$$Dcr = 1 - \frac{L_{pred}}{L_{longest}} \quad (9)$$

where L_c is the chain length from the source vertex to the current wave front vertex, L_d is the shortest length from the current wave front vertex to the destination vertex, as shown in Figure 6. The $L_{longest}$ can be seen as the length budget

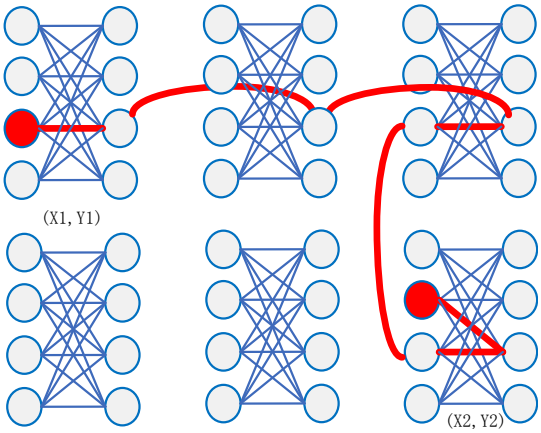

Figure 5: The Shortest Path

to route the current chain, since the routing result will not get worse if the L_c is shorter than the previous longest.

Compared to [14], we also modify the main routing cost function as the summation of the costs in order to balance their magnitude, as shown in Eq 7. The negotiation happens between two independent chains trying to use the same vertex, and the criticality eventually decides which chain takes the vertex and which chain should be detoured.

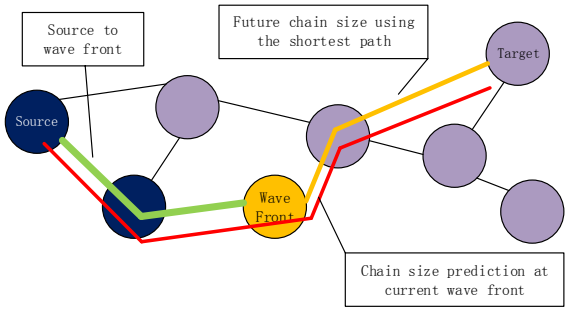

Figure 6: Chain Size Prediction

A bad routing result is often caused by the under estimation of the chain criticality. For example, a chain has low criticality in the last iteration will detour significantly in the new iteration. However, in the new iteration it may become the longest chain because of that.

Dynamic criticality uses prediction length to project its future criticality hence limits the chance of finding a worse result. At the beginning of the wave front propagation, the prediction length is short because the shortest path length contributes to the most of the path. As the wave keep propagating, the prediction length may increase due to the detour, and consequently the D_{cr} will also increase. the increased D_{cr} will encourage the router to use the short path rather than detour to reduce the congestion.

3.3 Rip-up and Reroute

The negotiation based routing dose not necessarily guarantee the optimal routing result. At the first several iterations, the vertexes in the routing graph is allowed to be shared by different chains. Then, the overflow is gradually solved by the increasing $C_s * P_i$. Ideally, the critical chains will take the shortest path, while others will be detoured.

However, if P_i is not properly designed, the routing may either too slow (P_i is too small) or the negotiation is not sufficient (P_i is too large). Both scenarios will leave the routing result sub-optimal.

To further improve the routing quality, we propose a rip-up and reroute stage, as described in Figure 7. Because the best routing solution is often hard to find, instead of trying to find the best solution, our strategy is to set a goal and then to find a solution that meets the goal.

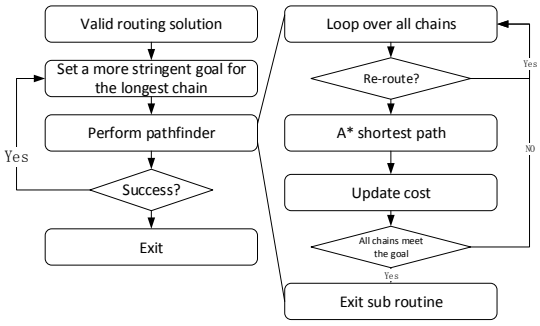

Figure 7: Rip-up and Reroute Flow

The rip-up and reroute starts from a valid routing result, which means that there is no vertex or edge shared by different chains. Then we set a limit on the longest chain length in the next iteration based on the longest chain length in the current iteration. Then we perform a similar negotiation-based routing to reroute some of the chains. A chain will be rerouted if meets the following criteria: (1) the chain uses a overflow vertexes or edges shared by other chains. (2) the chain violates the limit of the longest length.

At the beginning of the rip-up and reroute stage, there is no overflow vertexes or edges, so only those chains that violate the limit will be rerouted, and then some of the vertexes and edges will become overflow because of the rerouting. In the following iterations, the overflow will be gradually resolved as the penalty factor increases. Once a new result that has no overflow and do not violate the chain length limit is found, the negotiation routing will stop. Then, a more stringent limit will be set based on the newly found longest chain length and the procedures will be repeated. The stop condition for the rip-up and reroute is either the routing hits the iteration bounds or there is no room to improve as all the long chains are already taking the shortest path.

4. EXPERIMENTAL RESULTS

We chose CSP test cases from MCNC benchmark suite, as each combination circuit can be seen as a CSP problem where each constraint is a logic gate. Nevertheless, this work can be also applied to other non-circuit based CSP where each gate is replaced by a locally embedded constraint.

We implemented the proposed techniques in C++, and ran it over a Xeon-E5 2680 linux server with 64GB memory. As even the latest D-Wave device is too small, to better present the benefit of the proposed technique, we first conduct experiments by embedding the CSPs onto a hypothetical C100 Chimera architecture and compare it against the result from the previous research [5], which is denoted as QSAT. Secondly, we embed the test cases to the real D-Wave 2x quantum annealer, using the proposed techniques in the placement and routing.

4.1 C100 Chimera

In the first experiment, we embedded the CSPs onto a C100 Chimera architecture, which comprises 100 X 100 cells. The characteristics of the test cases are listed in the Table 2, following logic optimization performed by ABC synthesis tool. To accurately analyze and identify the effectiveness of our approach, we present the experimental result from placement and routing, respectively.

We compare the placement result in Table 2. The input of CSP is an netlist that represent a set of locally embedded constraints and their connections. By using our method, the placement run-time is improved by 1.8X on average along with reduction in the largest half perimeter wire length (HPWL). The result suggests our approach is much more efficient, despite an additional term in the cost function. This is achieved because our approach works in an incremental manner so that unnecessary and repetitive cost re-computation can be avoided.

We also compare the routing results in Table 3. For fair comparison, we started with the same placement result generated by the proposed placement algorithm, and compared routing using both our proposed techniques and the technique used in QSAT. Experimental results show that our approach greatly reduced the run-time by 1.8x to 3.3x and meanwhile is very effective in minimizing the longest chain length by 19% on average. We also notice that the result varies possibly because of the connection is very different in the original problem.

4.2 Ground State Probability Improvement

We report the performance improvement by embedding the CSPs onto the D-Wave 2x quantum annealer in Table 1. For each test case, we repeat the quantum annealing 10,000 times and gauge the state of the qubit. As the ground state energy can be calculated beforehand, using the summation of the lowest energy of each constraint and chain, we can find the probability of the ground state by counting the number of gauge that achieves the lowest energy. According to the experimental result, our approach improves the ground state probability by 29% on average.

Table 1: Embedding Experiment on D-Wave 2X

Design	The Longest Chain Size		Improvement	
	QSAT	FastEmbedding	TheLongestChain	P_{GS}
sct	52	36	30.77%	33.41%
b9	26	19	26.92%	39.39%
cordic	37	27	27.03%	27.72%
pcler8	58	41	29.31%	39.11%
parity	29	20	31.03%	22.96%
pcle	46	34	26.09%	21.97%
cc	62	44	29.03%	27.67%
cu	32	24	25.00%	31.24%
cm85a	30	23	23.33%	26.62%
x2	23	17	26.09%	28.33%
Average			27.46%	29.84%

5. CONCLUSIONS

EDA has witnessed a great success in assisting circuit design. In this work, we demonstrate an excellent example of using EDA techniques and philosophies to improve the performance of the D-Wave quantum annealer. As an emerging technology and a new computing paradigm, the quantum annealing is faced with many practical limitations. We see

Table 2: Placement Experiments on Architecture with 100 x 100 Cells

Design	GateNum	ChainNum	Placement					
			QSAT		FastEmbedding		Improvement	
			PlaceTime(S)	Max HWPL	PlaceTime(S)	Max HWPL	PlaceTime	Max HWPL
C6288	2370	2432	300.5	396	190.6	280	1.6X	29%
C5315	1775	2141	178.9	326	131.3	310	1.4X	5%
pair	1574	1918	164.1	294	99.1	213	1.7X	27%
dalu	1361	1509	118.3	149	77.1	125	1.5X	16%
frg2	1131	1421	83.8	98	54.5	81	1.5X	17%
C3540	1031	1129	73.3	162	50.3	160	1.5X	1%
i7	865	1261	63.4	60	39.8	57	1.6X	5%
x3	857	1125	49.2	91	37.7	81	1.3X	11%
apex6	786	1054	48.4	142	35.5	120	1.4X	15%
i9	736	910	41.9	120	27.0	112	1.6X	7%
Average							1.5X	13.4%

Table 3: Routing Experiments on Architecture with 100 x 100 Cells

Design	Routing					
	QSAT		FastEmbedding		Improvement	
	LongestChain	RouteTime(S)	LongestChain	RouteTime(S)	LongestChain	RouteTime
C6288	809	140.0	625	79.8	23%	1.8X
C5315	201	56.5	171	17.1	15%	3.3X
pair	495	51.2	364	27.4	26%	1.9X
dalu	447	75.0	436	26.3	3%	2.9X
frg2	326	42.4	261	21.0	20%	2.0X
C3540	273	65.8	221	24.0	19%	2.7X
i7	169	11.1	129	5.4	24%	2.1X
x3	137	17.2	130	8.4	5%	2.1X
apex6	220	13.0	157	5.6	29%	2.3X
i9	30	15.9	22	7.1	27%	2.2X
Average					19%	2.3X

that there is a strong demand to bring EDA techniques into the quantum computing area to address those challenges, as many EDA techniques are known to be effective in performing optimization under the presence of practical constraints.

In the future, we will continue our effort in exploring the new techniques to improve the performance of quantum annealer in solving CSP. Some of the future directions include: (1) performance driven constraint synthesis. (2) quantum annealer architecture exploration.

6. REFERENCES

[1] A. J. Berkley, M. W. Johnson, P. Bunyk, R. Harris, J. Johansson, T. Lanting, E. Ladizinsky, E. Tolkacheva, M. H. S. Amin, and G. Rose, "A scalable readout system for a superconducting adiabatic quantum optimization system," *Superconductor Science and Technology*, vol. 23, no. 10, p. 105014, Oct. 2010, arXiv: 0905.0891.

[2] N. G. Dickson, M. W. Johnson, M. H. Amin, R. Harris, F. Altomare, A. J. Berkley, P. Bunyk, J. Cai, E. M. Chapple, P. Chavez, F. Cioata, T. Cirip, P. deBuen, M. Drew-Brook, C. Enderud, S. Gildert, F. Hamze, J. P. Hilton, E. Hoskinson, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Lanting, T. Mahon, R. Neufeld, T. Oh, I. Perminov, C. Petroff, A. Przybysz, C. Rich, P. Spear, A. Tcaciuc, M. C. Thom, E. Tolkacheva, S. Uchaikin, J. Wang, A. B. Wilson, Z. Merali, and G. Rose, "Thermally assisted quantum annealing of a 16-qubit problem," *Nature Communications*, vol. 4, p. 1903, May 2013.

[3] S. H. Adachi and M. P. Henderson, "Application of Quantum Annealing to Training of Deep Neural Networks," *arXiv:1510.06356 [quant-ph, stat]*, Oct. 2015, arXiv: 1510.06356.

[4] A. Perdomo-Ortiz, N. Dickson, M. Drew-Brook, G. Rose, and A. Aspuru-Guzik, "Finding low-energy conformations of lattice protein models by quantum annealing," *arXiv:1204.5485 [quant-ph]*, Apr. 2012, arXiv: 1204.5485.

[5] J. Su, T. Tu, and L. He, "A quantum annealing approach for boolean satisfiability problem," in *Proceedings of the 53rd Annual Design Automation Conference*, ser. DAC '16. New York, NY, USA: ACM, 2016, pp. 148:1–148:6.

[6] G. Rosenberg, P. Haghnegahdar, P. Goddard, P. Carr, K. Wu, and M. L. de Prado, "Solving the Optimal Trading Trajectory Problem Using a Quantum Annealer," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 6, pp. 1053–1060, Sep. 2016, arXiv: 1508.06182.

[7] V. Choi, "Minor-embedding in adiabatic quantum computation: I. The parameter setting problem," *Quantum Information Processing*, vol. 7, no. 5, pp. 193–209, Oct. 2008.

[8] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos, "Faster parameterized algorithms for minor containment," *Theoretical Computer Science*, vol. 412, no. 50, pp. 7018–7028, Nov. 2011.

[9] J. Cai, W. G. Macready, and A. Roy, "A practical heuristic for finding graph minors," *arXiv:1406.2741 [quant-ph]*, Jun. 2014, arXiv: 1406.2741.

[10] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, and A. Roy, "Mapping constrained optimization problems to quantum annealing with application to fault diagnosis," *arXiv:1603.03111 [quant-ph]*, Mar. 2016, arXiv: 1603.03111.

[11] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. Macready, and A. Roy, "Discrete optimization using quantum annealing on sparse Ising models," *Interdisciplinary Physics*, vol. 2, p. 56, 2014.

[12] D. Venturelli, S. MandrÃă, S. Knysh, B. O'Gorman, R. Biswas, and V. Smelyanskiy, "Quantum Optimization of Fully-Connected Spin Glasses," *Physical Review X*, vol. 5, no. 3, Sep. 2015, arXiv: 1406.7553.

[13] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," in *Proceedings of the Third International ACM Symposium on Field-Programmable Gate Arrays, 1995. FPGA '95*, 1995, pp. 111–117.

[14] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. M. Fang, K. Kent, and J. Rose, "Vpr 5.0: Fpga cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 4, no. 4, pp. 32:1–32:23, Dec. 2011.