Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 635–639 ISSN 2300-5963 ACSIS, Vol. 43
DOI: 10.15439/2025F5496

Adiabatic Quantum Computing for the Subset Sum Problem: Preliminary Studies

C.A.F. Bernardes,* P. Castellucci,* D.S. Gonçalves,* E.I. Duzzioni* A. Mucherino,†

†Universidade Federal de Santa Catarina, Florianópolis, SC, Brazil

ces.aug.fre@gmail.com, pedro.castellucci@ufsc.br,

douglas.goncalves@ufsc.br, eduardo.duzzioni@ufsc.br

†IRISA, Université de Rennes, Rennes, France

antonio.mucherino@irisa.fr

***Abstract*—The Subset Sum Problem (SSP) is one of those combinatorial problems that are very easy to understand (take a bunch of integer numbers and verify whether there exists a subset of these numbers which sums up to a given target integer), but it can be very difficult to solve. The SSP is actually an NP-complete problem, but it is "weakly" NP-hard, implying that there are instances of SSP that can be solved in polynomial time. For this particular problem, the instance hardness can be measured by evaluating the so-called "density" index, which basically compares the number of involved integer numbers to the number of bits we need for their binary representation. In our preliminary study on the use of adiabatic quantum computing for the SSP, we investigate the actual feasibility in solving hard instances of the problem. In fact, hard SSP instances are those requiring a large number of bits for the representation of the integers, while the analog nature of the quantum computer does not allow us to ensure highly accurate integer representations. Some preliminary computational experiments performed on D-Wave quantum annealer are presented and compared to standard solvers for classical computers.**

I. Introduction

GIVEN a set $\mathcal{A}$ of n positive integer numbers and a positive target t, the Subset Sum Problem (SSP) asks whether there is a subset $\mathcal{S}$ of $\mathcal{A}$ such that the elements of $\mathcal{S}$ sum up exactly to t. The original set $\mathcal{A}$ may be a *multiset*, but in this work we assume, for simplicity, that it contains no repeated elements. In a typical problem variant, the SSP asks to enumerate *all* possible solutions (i.e. all subsets) of a given SSP instance. But again, for simplicity, we make the assumption in this work that only one solution is required.

The SSP is one of the famous Karp's NP-complete problems [12]. It is considered to be *weakly* NP-hard, because there exist large classes of SSP instances for which algorithms admitting a pseudo-polynomial complexity have been proposed [6]. In order to find out the actual hardness of a given SSP instance, the so-called *density* index can be employed:

$$\rho := \frac{n}{\ell(n)},$$

where

$$\ell(n) = \log_2 \max_{a \in \mathcal{A}} a.$$

SSP instances having a density index ρ equal to 1 are the hardest to solve, and indeed, to date, no pseudo-polynomial algorithm for these instances have been found. In these instances, the number of bits necessary to represent the integers in $\mathcal{A}$ equals the number of integers in $\mathcal{A}$. In this work, we will focus our attention on these density-1 instances.

There is a large scientific literature on the SSP. A large review of the existing literature is out of the scope of this short paper, so that we will limit ourselves to cite only a few recent works aiming at improving the performance of SSP algorithms for hard "density-1" instances. The algorithm that has been considered to be the *fastest* for these instances for a very long time is one proposed by Horowitz and Sahni in 1974 [11], which has, however, an exponential theoretical complexity. In more recent times, the works presented in [3] and [19], for example, showed that there exist algorithmic solutions which can exhibit better performance. It is important to remark that the research is progressing in parallel in both areas of deterministic algorithms (such as in [3]) and heuristics (such as in [19]). The most recent effort in the former category of algorithms was presented in the pre-print [18]. To date, however, algorithms having polynomial complexity have not been found yet for solving density-1 instances (a discovery that could imply that **P**=**NP**, which is unlikely).

Many researchers are exploring nowadays alternative approaches for tackling the SSP. Some of us are for example currently working on a novel approach in optical computing [10], where the use of the spatial properties of light allow us to speedup the solution of hard SSP instances (very recently, moreover, the same optical circuit has been adapted to perform single-key cryptography in [7]). Even if these optical circuits are not "quantum devices", they exploit physical phenomena normally used in quantum computing, such as interference.

Some recent theoretical studies aiming at exploiting the potential "quantum advantage" of modern quantum technologies for solving the SSP can be found for example in [2], [5], [9].

In this work, we study the feasibility of solving hard density-1 instances of the SSP by adiabatic quantum computing. To the best of our knowledge, this alternative approach to the SSP was only a little studied in recent times. One example of such studies can be found in [20], where a mathematical model similar to ours is introduced, whose solution is however attempted in a different manner. Moreover, in our study, we will focus our attention on the analog nature of the considered quantum computer, and on the consequent representation issues for the elements of $\mathcal{A}$ (as well as for the target t). We also point out that several of the publications cited above are mostly theoretical and provide no computational experiments.

The rest of the paper is organized as follows. Two general methods for the SSP are briefly presented in Section II, because they will be involved in the performance comparisons presented with our computational experiments. Section III will briefly describe the physics behind the quantum annealer implemented by the D-Wave quantum processing units, and propose a reformulation of the SSP as a Quadratic Unconstrained Binary Optimization (QUBO), a problem format that can handled by D-Wave quantum computers. Finally, our experiments will be reported in Section IV, and Section V will conclude the paper.

II. Methods for the SSP

A. The BP algorithm

We begin this section by presenting a generic algorithm whose worst-case complexity is exponential, but that is suitable for all kinds of SSP instances, including the hard ones. This algorithm is presented in different guises in textbooks about NP-complete problems, and under different names. We make the choice to refer to it as the Branch-and-Prune (BP) algorithm [15]: this is the name of an algorithm for a more general problem, the Discretizable Distance Geometry Problem (DDGP) [17], which can be reduced to the SSP under special conditions. The DDGP has several important applications in various research fields [16].

The BP algorithm is based on the idea to construct a binary tree, where every path from the root to one of the leaf nodes represents a possible subset $\mathcal{S}$ of the original set $\mathcal{A}$. Every layer of the tree is related to one of the integers in $\mathcal{A}$: if the subsets "opting" for the left-handed branch at a given level do not contain the corresponding integer, then this same integer would instead be contained in all other subsets where the right-handed branch is selected. The tree grows regularly by doubling its nodes as we step from one layer to the next. The full tree contains therefore 2^n nodes, but they do not need to be all explored. Some tree branches can in fact be *pruned* by verifying some feasibility tests. If the integers that are selected in the current branch sum up to a value that is larger than the target t, for example, then we can prune the current branch. Also, if the sum of all remaining (i.e. not yet considered) integers at the current tree layer does not allow the current partial sum to reach the desired target t, then the branch can also be pruned.

An implementation in C programming language of the BP algorithm for the SSP is publicly available on the GitHub[1].

B. MILP formulation

Given the set $\mathcal{A} = \{a_1, \ldots, a_n\}$ and the target t, the SSP can be cast as the following boolean linear equation:

$$\sum_{i=1}^{n} a_i x_i = t, \tag{1}$$

where $x \in \{0,1\}^n$. Since, in principle, the SSP instance at hand may be infeasible, we need to take into consideration the case where the equation above may admit no solutions. For this reason, we introduce a slack variable $z \in \mathbb{R}$ and formulate the following optimization problem:

$$\begin{aligned} \min_{x,z} \quad & |z| \\ \text{s.t} \quad & \sum_{i=1}^{n} a_i x_i + z = t, \\ & x \in \{0,1\}^n, z \in \mathbb{R}. \end{aligned}$$

If the optimal value for the problem above is zero, then the SSP has a solution given by the binary decision variables x_i, where $x_i = 1$ indicates that a_i is a member of the solution subset $\mathcal{S}$.

In order to make the objective function linear, we can represent $z = \Delta^+ - \Delta^-$, where Δ^+, Δ^- are non-negative real variables. Then, we have the following Mixed Integer Linear Program (MILP):

$$\begin{aligned} \min_{x,\Delta^+,\Delta^-} \quad & \Delta^+ + \Delta^- \\ \text{s.t} \quad & a^T x + \Delta^+ - \Delta^- = t, \\ & \Delta^+ \geq 0, \quad \Delta^- \geq 0, \\ & x_i \in \{0,1\}, \quad i = 1, \ldots, n, \end{aligned} \tag{2}$$

where

$$a^T x = \sum_{i=1}^{n} a_i x_i.$$

Such a formulation is ideal for MILP solvers (as for example Gurobi [22], which we will use in our computational experiments) that implement branch-and-bound techniques for MILPs.

III. Quantum Annealing

Quantum annealing is an optimization technique used to find the ground state of a given Hamiltonian through the adiabatic evolution of the quantum system [1]. The problem to be solved can be encoded in the time-dependent Hamiltonian:

$$H(\tau) = A(\tau)H_I + B(\tau)H_F,$$

where the time-dependent coefficients $A(\tau)$ and $B(\tau)$ satisfy the condition $A(0) \neq 0$, $B(0) = 0$ and $B(T) \neq 0$, $A(T) =$

[1] https://github.com/mucherino/DistanceGeometry

0, with T being the total evolution time, which corresponds to the computation time. Following the proposal of quantum annealing in the transverse field Ising model [13], the initial Hamiltonian for n qubits is described by

$$H_I = -\sum_i^n \sigma_x^i$$

and the final one by

$$H_F = \sum_i^n h_i \sigma_z^i + \sum_{i,j}^n J_{i,j} \sigma_z^i \sigma_z^j.$$

In these equations, h_i denotes the bias field acting on the i-th qubit and $J_{i,j}$ is the coupling constant between the qubits i and j. Besides of being an easy-to-prepare Hamiltonian, whose ground state is

$$|\psi(0)\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes n},$$

H_I acts as a driver Hamiltonian, which is responsible for mixing the states in the computational basis. Therefore, if initially the physical system starts in the ground state of the Hamiltonian H_I and is adiabatically driven until the final configuration determined by H_F, then, by performing measurements in the computational basis, we can find the solution to our problem as a string of bits.

The strategy described above is the one implemented by the quantum computers produced by the D-Wave Systems Inc. company. In practice, the quantum system is working on 20 mK [21], so there is a no null probability of finding the system outside the ground state, indicating the probabilistic character of the solution of the computation. This implies that, in order to solve a given problem using the D-Wave's quantum processing units, it is necessary to run the calculations several times.

Instead of formulating the problem in terms of the Ising Hamiltonian, it is possible to write it in the form of a Quadratic Unconstrained Binary Optimization (QUBO) problem [8]. The mapping of Pauli operators (σ^i) to QUBO operators (q^i) is obtained by $\sigma^i = 2q^i - I$, where I is the identity matrix. Therefore, the QUBO Hamiltonian becomes:

$$H_F = \sum_i^n Q_i q^i + \sum_{i,j}^n Q_{i,j} q^i q^j,$$

where the operators q^i have eigenvalues 0 or 1, provided that the eigenvalues of the Pauli matrices are -1 and 1. From a practical point of view, the problem is in general written in the QUBO formulation and then translated to the Ising model, which is the natural physical formulation.

In order to obtain a QUBO formulation for the SSP, we remark that $x \in \{0,1\}^n$ solves equ. (1) if it is a zero for the quadratic function:

$$q(x) = (a^T x - t)^2, \tag{3}$$

where a is a column vector with elements a_i, and the same goes for $x \in \{0,1\}^n$. We can therefore re-write:

$$q(x) = x^T a a^T x - (2ta)^T x + t^2 = x^T Q x + b^T x + c,$$

with $Q = aa^T$, $b = -2ta$ and $c = t^2$. By ignoring the constant term c, we have a correspondence with the QUBO Hamiltonian where

$$Q_i = -2ta_i, \quad Q_{i,j} = a_i a_j,$$

are its coefficients.

IV. Computational experiments

A. Generating hard SSP instances

In order to generate hard SSP instances, we make sure that its density index is as close as possible to 1. To this end, we define the very first integer to be included in the original set $\mathcal{A}$ as 2^n, where n is the desired size of the instance. All other integers, used to fill the set $\mathcal{A}$ until it reaches the desired size n, are generated by randomly choosing integers with a uniform distribution in the (open) interval $(1, 2^n)$.

In order to control the hardness of our instances, we also follow some additional recommendations given in [4] when generating our instances. The range of the target t is constrained and taken between

$$(n/2 - \sqrt{n}) \cdot 2^{n-1} \quad \text{and} \quad (n/2 + \sqrt{n}) \cdot 2^{n-1},$$

while approximately half of the original integers in $\mathcal{A}$ are chosen to take part to one pre-constructed SSP solution, which needs to satisfy the selected target t.

Our method for a random generation of hard instances begins by selecting the maximum integer in the set, as described above, and then selects the target t in the bounds given above. Only at this point does the random generation of the other integers begin, and the decision on whether or not to include each generated integer (with chance 1/2) in one associated solution is randomly made. However, the probability of defining a valid solution in this way is extremely unlikely (the randomly chosen integers should sum up exactly at t), and therefore a *correction* step is implemented in our generation algorithm, where the values of the integers selected to form an SSP solution are modified in order to have their sum correspond to the pre-defined target t.

Before delivering the generated instance, our method double-checks that the number of integers forming the constructed SSP solution actually concerns exactly the half of the integers in $\mathcal{A}$ (the only tolerated approximation is the one implied by an odd n). When this is not the case, integers belonging to the SSP solution are either randomly split or fusioned, while other integers (not belonging to the solution) may be erased or added in order to keep the cardinality of $\mathcal{A}$ equal to n.

TABLE I
SOME COMPUTATIONAL EXPERIMENTS WHERE THE GENERATED SSP INSTANCES ARE SOLVED BY A C IMPLEMENTATION OF THE BP ALGORITHM, BY GUROBI (MILP MODEL), BY THE SA SIMULATION PROVIDED BY D-WAVE, AND FINALLY BY D-WAVE ITSELF. THE VALUES OF THE TARGETS, FOR THE 4 INSTANCES, ARE $t_1 = 1024$, $t_2 = 3145728$, $t_3 = 5368709120$, AND $t_4 = 7696581394432$. THE VALUE OF ϵ INDICATES THE OBSERVED ERROR WITH THE RESPECT TO THE KNOWN SOLUTIONS. THE COMPUTATIONAL TIME IS GIVEN IN SECONDS.

Size/target	10	t_1	20	t_2	30	t_3	40	t_4
	ϵ	time	ϵ	time	ϵ	time	ϵ	time
BP	0	0.00	0	0.00	0	0.07	0	9.36
Gurobi	0	0.01	0	0.31	0	1.53	0	1138.00
SA	0	2.00	-8	4.00	-249	8.00	21959	15.00
D-Wave	0	0.02	11	0.25	20926	1.1	173294	2.80

B. Preliminary experiments

We provide some preliminary computational experiments where we solve 4 hard instances of the SSP, having 4 different sizes: 10, 20, 30 and 40. We limit ourselves to instances having size 40 because we have soon noticed that we had already hit the limits of the quantum computer when using instances having this relatively small size. Since our instances all have density equal to 1, the number of bits necessary to represent the elements of $\mathcal{A}$ in the 4 instances is 10, 20, 30 and 40, respectively. On standard 64-bit machines, therefore, standard primitive types, in the majority of programming languages, can hold values that are sufficiently large for our purposes. This is, however, not the case in the analog environment of D-Wave QPUs (Quantum Processing Units).

Our experiments are reported in Table I. The value of ϵ indicates the absolute error with respect to the known solutions, while the computational time is given in seconds. All experiments have been performed on a laptop computer equipped with an 11^{th} Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz, running Linux. We used Gurobi version 11, with default parameters except for `IntFeasTol` $= 10^{-9}$ and `NumericFocus` $= 1$. Concerning D-Wave QPUs, we considered two topologies, ZEPHYR and CHIMERA, and varied `num_reads` in $[2000, 4000]$ and `annealing_time` in $[40, 80]$. The results reported in Table I are the best over all the variations considered and over 10 runs/jobs.

We can remark that the BP algorithm, in spite of its worst-case exponential complexity, performs quite efficiently on a standard computer: it is able to solve *exactly* all SSP instances in a reasonable time. Gurobi applied to (2) also solved all instances but its drawback is the computation time. The Simulated Annealing (SA) meta-heuristics [14], implemented in the D-Wave standard Python package, appears to be slower than BP and faster than Gurobi, and provides solutions affected by large errors. This tendency is then amplified when running the experiments on D-Wave. The analog quantum computer is able to output its solution quickly (it is even faster than BP on the largest instances), but the given solutions are affected by much larger errors.

One interesting remark is that most of the solutions found by D-Wave are "local" minimizers, in the following sense. The concept of neighbouring, and hence the one of *locality*, makes sense in the context of the SSP only if we take into account the discrete nature of the search space. When we talk about local solutions, we make reference here to solutions that can be obtained from a provided solution x by performing only one bit flip: in other words, the solutions that are considered to be in the neighbourhood of x are those for which only one integer is added or removed from the current subset. We consider all D-Wave solutions to be local minimizers because they cannot be improved (with respect to the loss function (3)) by one single bit flip.

On D-Wave QPUs, the biases h_i and coupling parameters J_{ij} have specific physical ranges that the hardware can implement. These are typically floating-point numbers rather than integers, and they are restricted to a relatively small range: typically $h_i \in [-2.0, 2.0]$ and $J_{ij} \in [-1.0, 1.0]$. Besides, the D-Wave software interface has an automatic scaling feature which scales the problem data in order to fit in these intervals. Thus, if the problem data has a very large range, scaling them all down can make the smaller differences less distinguishable by the noisy analog hardware, potentially affecting solution quality.

We conjecture that, when D-Wave is actually not able to find the correct solutions, it is because of the representation errors implied by its analog nature. Part of our future works will be devoted to further studying this conjecture.

V. CONCLUSIONS

This paper focuses on the SSP, one of the most important, well-known and largely studied NP-complete problems. We have briefly surveyed two basic methods for the solution of SSP instances, provided a QUBO formulation of the problem, which we have then used for performing some preliminary computational experiments on D-Wave.

The main point in this work is the following: can we actually represent hard SSP instances on the quantum computer? Or is this rather possible only for some of the "easy" SSP instances? Our experiments show that an important difficulty is related to the accuracy in the representation of the integers in the set $\mathcal{A}$. As the number of bits necessary for representing the integers increases, it becomes harder and harder to have an error-free representation in the analog quantum computer. Our preliminary experiments seem to indicate that the primary impediment to quantum advantage in this domain is not algorithmic design but rather the limitations in representation in the analog computer.

Work is currently in progress to design methods for overcoming this representation issue. We will also study the possibility to represent and solve more general problems on D-Wave, such as for example the DDGP mentioned in the Introduction.

ACKNOWLEDGMENTS

This work was partially supported by the ANR project EVARISTE, which financed in May 2025 a visit of AM to the other co-authors at the Federal University of Santa Catarina. This paper is the result of the work performed during the visit. The authors are also grateful to the 5 reviewers for their fruitful comments.

REFERENCES

[1] T. Albash, D.A. Lidar, *Adiabatic Quantum Computation*, Reviews of Modern Physics **90**(1), 015002, 2018.

[2] J. Allcock, Y. Hamoudi, A. Joux, F. Klingelhöfer, M. Santha, *Classical and Quantum Algorithms for Variants of Subset-Sum via Dynamic Programming*, 30^{th} Annual European Symposium on Algorithms (ESA22), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 28 pages, 2022.

[3] P. Austrin, P. Kaski, M. Koivisto, J. Nederlof, *Dense Subset Sum May Be the Hardest*, Leibniz International Proceedings in Informatics, 33^{rd} Symposium on Theoretical Aspects of Computer Science, Article No. 13, 13:1–13:14, 2016.

[4] D.J. Bernstein, S. Jeffery, T. Lange, A. Meurer, *Quantum Algorithms for the Subset Sum Problem*. In: Post-Quantum Cryptography, Proceedings 5, 16–33, 5^{th} International Workshop PQCrypto 2013.

[5] X. Bonnetain, R. Bricout, A. Schrottenloher, Y. Shen, *Improved Classical and Quantum Algorithms for Subset-Sum*, International Conference on the Theory and Application of Cryptology and Information Security, Springer International Publishing, 633–666, 2020.

[6] M.J. Coster, A. Joux, B.A. LaMacchia, A.M. Odlyzko, C.P. Schnorr, J. Stern, *Improved Low-density Subset Sum Algorithms*, Computational Complexity **2**, 111–128, 1992.

[7] M.G. Damaceno, A. Mucherino, R. Medeiros de Araújo, P.H. Souto Ribeiro, N. Rubiano da Silva, *Experimental Investigation of Optical Processing With Spatial Light Modulation*, arXiv preprint arXiv:2507.03821, 2025.

[8] F. Glover, G. Kochenberger, Y. Du, *A Tutorial on Formulating and Using QUBO Models*, arXiv preprint arXiv:1811.11538, 2018.

[9] A. Helm, A. May, *The Power of few Qubits and Collisions–Subset Sum below Grover's Bound*, Springer International Publishing, International Conference on Post-Quantum Cryptography, 445–460. 2020.

[10] S.B. Hengeveld, N. Rubiano da Silva, D.S. Gonçalves, P.H. Souto Ribeiro, A. Mucherino, *An Optical Processor for Matrix-by-Vector Multiplication: an Application to the Distance Geometry Problem in 1D*, Journal of Optics **24**(1), 015701, 2022.

[11] E. Horowitz, S. Sahni, *Computing Partitions with Applications to the Knapsack Problem*, Journal of the ACM (JACM) **21**(2), 277–292, 1974.

[12] R.M. Karp, *Reducibility Among Combinatorial Problems*. In: R.E. Miller, J.W. Thatcher, J.D. Bohlinger (Eds.), "Complexity of Computer Computations". New York: Plenum. 85–103, 1972.

[13] T. Kadowaki, H. Nishimori, *Quantum Annealing in the Transverse Ising Model*, Physical Review E, **58**(5), 5355, 1998.

[14] S. Kirkpatrick, C.D. Gelatt, M.P Vecchi, *Optimization by Simulated Annealing*, Science **220**, 671–680, 1983.

[15] L. Liberti, C. Lavor, N. Maculan, *A Branch-and-Prune Algorithm for the Molecular Distance Geometry Problem*, International Transactions in Operational Research **15**, 1–17, 2008.

[16] L. Liberti, C. Lavor, N. Maculan, A. Mucherino, *Euclidean Distance Geometry and Applications*, SIAM Review **56**(1), 3–69, 2014.

[17] A. Mucherino, C. Lavor, L. Liberti, *The Discretizable Distance Geometry Problem*, Optimization Letters **6**(8), 1671–1686, 2012.

[18] J. Salas, *Beyond Worst-Case Subset Sum: An Adaptive, Structure-Aware Solver with Sub-$2^{n/2}$ Enumeration*, arXiv preprint arXiv:2503.20162, 2025.

[19] C.P. Schnorr, T. Shevchenko, *Solving Subset Sum Problems of Density close to 1 by Randomized BKZ-Reduction*, Cryptology ePrint Archive, 5 pages, 2012.

[20] Q. Zheng, Y. Miaomiao, Z. Pingyu, W. Yan, L. Weihong, X. Ping, *Solving the Subset Sum Problem by the Quantum Ising Model with Variational Quantum Optimization based on Conditional Values at Risk*, Science China Physics, Mechanics & Astronomy **67**(8), 280311, 2024.

[21] D-Wave Systems Inc., *Technical Description of the D-Wave Quantum Processing Unit*, `https://docs.dwavesys.com`, 2021.

[22] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, `https://www.gurobi.com`, 2024.