Application of nature inspired metaheuristics to the optimization of QAOA circuits.

January 7, 2022

Abstract

abstract

1 Introduction

Metaheuristics are not a novel method for optimization problems. In fact, one of the first metaheuristics was proposed in 1982 with the one called simulated annealing [1]. A metaheuristic is simply a procedure that coordinates the use of heuristics to produce high quality solutions [9]. Where an heuristic is: 'a method which, on the basis of experience or judgement, seems likely to yield a reasonable solution to a problem, but which cannot be guaranteed to produce the mathematically optimal solution.'[9].

Metaheuristics are composed of two phases: exploration and exploitation [1]. In the exploration phase, the metaheuristic searches the solution space of the objective function. While in the exploitation phase the candidates solutions, obtained from the previous phase, are evaluated to find the one with the best quality. Additionally, metaheuristics have several controllable parameters for flexibility, although, it requires the careful configuration of these parameters. [9]

Among the different types of metaheuristics, there is a particular group that will be of interest later on. This group is the population based methods. Some examples of these are: Ant colony optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC) and Bat optimization (BA) [1].

On the other hand, the quantum approximation optimization algorithm is a method to produce approximate solutions to combinatorial optimization problems[5]. The algorithm depends on a parameter, usually called p and the algorithm gets a better solution the bigger the parameter is. This method uses a set of parametrized unitary transformations. To get the right parameters of these transformations, it is a common practice to use a classical optimizers. So the problem that we'll address is to find the performance of some metaheuristics in this optimization process.

2 nature inspired Metaheuristics

We will consider four metaheuristics for the problem at hand. The chosen metaheuristics are: Bat optimization(BAT), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC) and Ant colony optimization (ACO). This methods were selected because they have shown good results with other optimization problems, like the optimization of weights for neural networks [8] [7] [1].

2.1 Bat optimization

The key feature of this method is exploiting the fact that usually bats use echolocation to hunt their prey [3]. Echolocation happens when bats emit a sound and then they listen to the bounced sound to identify their prey [3]. And even further, studies show that bats which use echolocation can create a three dimensional scenario of their surrounding, the moving speed of the prey, the distance of the target, among other things [3]. As such, the bat optimization method can be summarized as follows:

Algorithm 1 Bat optimization

procedure BA
    bat population ← initialize a list of bats
    for each bat in bat population ← initialize its pulse rate r, loudness A,
    velocity, initial position, pulse interval and frequency (min and max)
    $t \leftarrow 0$
    while (t < number of iterations):
        best position ← best position in the population
        average loudness ← get average loudness from population
        for each bat in population:
            frequency ← frequency min +
            (frequency max-frequency min)*random number
            velocity ← velocity + (position - best position)*current frequency
            position ← position + velocity
            if random number > current pulse interval **then**
                position ← best position+random number*average loudness
            position ← random position+random number*average loudness
            if random number < loudness and cost < best position cost **then**
                loudness ← random number*loudness
                pulse interval ← initial pulse interval $* (1 - (e^{-t*random}))$
    Return the best position found by the bats

2.2 Particle Swarm Optimization

While bat optimization technique uses bats and their corresponding echolocation to search the solution space, the particle swarm optimization uses the swarm behaviour of particles to optimize an objective function. The swarm behaviour as outlined in

[8] must accomplish 5 points. Some of these points are: the swarm must be able to make simple space and time calculations, the swarm must be able to respond to quality factors in the environment, the swarm should not change its mode of behavior every time the environment changes and the swarm must be able to change behavior mode when it's worth the computational price [8]. The key idea behind the algorithm is that the swarm may follow the direction of the best solution found, although the particles may not follow this direction accordingly to a random variable. Additionally, a particle adjust it's velocity accordingly to the proximity of its current objective. The method can be summarized as follows:

Algorithm 2 Particle swarm optimization

```
procedure PSO
    swarm ← initialize a list of particles
    for each particle in swarm ← initialize its initial position, velocity
    and best position found
    w ← number in range 0 < w < 1
    c1 ← number in range 0 < c1 < 1
    c2 ← number in range 0 < c2 < 1
    while (t < number of iterations):
        best position ← best position in the swarm
        for each particle in population:
            current position ← current position+velocity
            if current position cost < best position found cost then
                best position found ← current position
            velocity ← w*velocity+
            c1*random number*(best position found-current position)+
            c2*random number*(best position found by swarm-current position)
    Return the best position found by the swarm
```

2.3 Ant colony optimization

Originally ACO was proposed to solve discrete optimization problems [9]. However, there are proposals to change the nature of the optimizer to one that solves continuous optimization problems. We'll use one of these proposals. Before explaining the method, we'll explain the key feature of the ant colony optimization. Ants communicate with other ants in the colony by a secretion called pheromone [10]. This chemical is produce when the ants in the colony move to the candidate solutions and then they return to the colony. Usually the pheromone trail to indicate the quality of the solution to other ants. Each ants that passes over the pheromone trail reinforces the trail and in consequence, the trail will be more appealing to other ants [9] [2]. Even further, this pheromone trail has the characteristic that evaporates over time. This is a mechanism that allows the ants to forget bad solution and explore new ones [9]. The general method can be summarized as follows:

Algorithm 3 Ant colony optimization

```
procedure ACO
    colony ← initialize a list of ants
    points ← initialize a list of uniform distributed points
    for each ant in colony ← initialize its memory, memory limit and current location
    while (t < number of iterations):
        for each ant in colony:
            if random number > q then
                ant current location ← best position
                store current position in ant memory
                move the ant location accordingly to a gradient optimizer
            else
                ant current location ← grab a random point with a probability of
                pointPheromone/totalPheromone
                store current position in ant memory
                move the ant location accordingly to a gradient optimizer
        for the best ant in colony:
            current location pheromone ← pheromone + (1/error)
        for each point:
            point pheromone ← pheromone ∗ (1 − p)
    Return the best position found by the colony
```

2.4 Artificial bee colony optimization

In the bee colony optimization technique we use bees as an agent to solve the optimization problem. In the proposed method by [7] bees can be one of three types: The worker bees, the onlooker bees and the scout bees. The worker bees go to the position of a potential solution and explore the neighborhood to see if a better solution can be found. The onlooker bees search the best proposals of worker bees accordingly to a random decision and finally the scout bees explore a new solution at random. This interaction can be summarized by the following pseudocode:

Algorithm 4 Artificial Bee Colony optimization

```
procedure ABC
    swarm ← initialize a list of worker bees
    onlookers ← initialize a list of onlooker bees
    for each bee in swarm ← initialize its current location at random
    for each bee in onlookers ← initialize its current location at random
    while (t < number of iterations):
        for each bee in swarm:
            new position ← bee position +
            random(-a,a)*(bee position[random] - swarm[random].position[random])
            if new position cost <  bee position cost then
                bee position ← new position
            else
                if solution has not improved in r tries then
                    bee position ← lower bound +
                     random uniform(0, 1)*(upper bound-lower bound)
        for each bee in onlookers:
            selected position ← select a position of the bees in swarm using a roulette selection
            new position ← selected position +
            random(-a,a)*(selected position[random] - swarm[random].position[random])
            if new position cost <  bee position cost then
                selected position ← new position
    Return the best position found by the colony
```

3 Quantum Approximation Optimization Algorithm

Recently, a new way to address NP-Complete problems has been created. This new method is the quantum approximate optimization algorithm (QAOA). This method consists in exploiting the properties of quantum mechanics to arrive to an approximation to a solution for a NP-Complete problem. The general method can be summarized as follows:

1. Create an initial state that is a uniform superposition over the computational basis states.
2. Apply the unitary transformations $U(\boldsymbol{\gamma}) = e^{-i\gamma}$ and $U(\boldsymbol{\beta}) = e^{-i\beta H_B}$, p times. Where β and α are a set of angles, H_P is the problem Hamiltonian and H_B is the mixing Hamiltonian.
3. Measure in the computational basis and calculate the cost.

After calculating the cost in step 3, it is common to use an optimizer to change the value of the angles in order to get a better result.

Additionally, one form of measuring the overall performance of a QAOA circuit is by calculating the approximation ratio. The approximation ratio is calculated by diving the probability of the solution by the probability of the highest output [6].

3.1 Max-cut

The max-cut is a well know problem in computer science. It consists in finding two disjoint sets in a graph such that the number of edges between nodes in different sets are maximized [5]. For this problem, the proposed mixing Hamiltonian and problem Hamiltonian are taken directly from [6]:

$H_P = \sum_{(i,j)\in E} \frac{1}{2}\left(1 - Z_i Z_j\right)$

$H_B = \sum_{(j=1}^{n} \sigma_j^x$

4 Experimental study

To evaluate the performance of the QAOA circuits while using the nature inspired optimizers we'll compare the execution time, approximation ratio and probability of the solution with other common optimization techniques such as Nelder-Mead, COBYLA and SLSQP optimizers. The selected problem to compare the performance is the well know max cut problem.

4.1 Max-cut

The experiments for the max-cut problem consisted in creating 4-complete to 8 complete graphs. Afterwards, we executed the QAOA instance using a qasm simulator to check the execution time, approximation ratio and probability of the solution. Additionally, for the qasm simulator test, we decided to get the output of the circuit with the same parameters 20 times and get the mean of the probabilities to get a better result.

The following hyper parameters were used for the optimizers:

Method	Hyperparameters
PSO	particles=20, w=0.4, c1=0.1, c2=0.1, iterations=50
BA	bats=5, iterations=50, alfa= 0.4, gamma=0.4
ACO	points=100, ants=20, q=0.01, evaporation rate=0.9, iterations = 100
ABC	points=100, ants=20, q=0.01, evaporation rate=0.9, iterations = 100
Nelder-Mead	Absolute error in between iterations that is acceptable for convergence=0.0001, maximum iterations=until convergence.
COBYLA	rhobeg= 1.0, maxiter= 1000
SLSQP	maxiter= 100, ftol= 1e-06, eps= 1.4901161193847656e-08

Table 1: Hyperparameters for the optimizers

5 Results

In the following section, we'll present the results of the executions described in the experimental study section.

5.1 Max-cut

6 Conclusions

References

[1] Ashraf Mohamed Hemeida, Somaia Awad Hassan, Al-Attar Ali Mohamed, Salem Alkhalaf, Mountasser Mohamed Mahmoud, Tomonobu Senjyu, Ayman Bahaa El-Din, Nature-inspired algorithms for feed-forward neural network classifiers: A survey of one decade of research, Ain Shams Engineering Journal, Volume 11, Issue 3, 2020, Pages 659-675, ISSN 2090-4479, https://doi.org/10.1016/j.asej.2020.01.007.

[2] Mavrovouniotis, M., Yang, S. Training neural networks with ant colony optimization algorithms for pattern classification. Soft Comput 19, 1511–1522 (2015). https://doi.org/10.1007/s00500-014-1334-5

[3] Yang, Xin-She. (2010). A New Metaheuristic Bat-Inspired Algorithm. 284. 10.1007/978-3-642-12538-6_6.

[4] Grand'rive, Pierre & Hullo, Jean-Francois. (2019). Knapsack Problem variants of QAOA for battery revenue optimisation.

[5] Edward Farhi and Jeffrey Goldstone and Sam Gutmann. (2014). A Quantum Approximate Optimization Algorithm.

[6] Ryan Hoque. (2020). The Quantum Approximate Optimization Algorithm.

[7] Dervis Karaboga (2010) Artificial bee colony algorithm. Scholarpedia, 5(3):6915.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

[9] Silver, E. An overview of heuristic solution methods. J Oper Res Soc 55, 936–956 (2004). https://doi.org/10.1057/palgrave.jors.2601758

[10] Yang, X.-S.: Nature-inspired Metaheuristic Algorithms. Luniver Press, (2008).