Simulated Annealing for the Number Partitioning Problem

Kaveri Priya Putti

Abstract

Number partitioning - also referred to as the number bi-partitioning problem, or the 2-partition problem - is one of Karp's original NP-hard problems, and can be stated as follows. We have a set S of n numbers, and the goal of the optimisation problem is separate them into two disjoint subsets S1 and S2 such that the difference of the sums of the two subsets is minimised.

I. Algorithm

Simulated annealing is a classical meta-heuristic to find approximate solutions to optimisation problems in a large search space. It uses stochastic or Monte Carlo sampling to mimic the way that materials cool down to stable and ordered configurations (Kirkpatrick et al). The following pseudocode presents the simulated annealing heuristic:

```
procedure SIMULATED-ANNEALING returns A STATE s_k
    inputs: T_0, initial temperature
            J, cost function
            s_0, initial state
            temp-schedule, cooling schedule
            neighbor, neighbor state function
    T ← T_0
    s_k ← s_0
    for t = 1 to t_max do
        s_{k+1} ← neighbor(s_k)
        ΔE ← J(s_{k+1}) − J(s_k)
        if min(1, e^{−ΔE/T}) ⩾ rand(0, 1) then
            s_k ← s_{k+1}
        end if
        T ← temp-schedule(t)
    end for
end procedure
```

II. Approach and Analysis

While coding out the solution for the number partitioning problem with SA, I found that the main components of the algorithm include the following:

1. It starts from a state $state_0$ and continues to either a maximum of *iterations* or until a state where $cost = 0$ is found.
2. The cost function is defined and computed based on the paper "Ising formulations of many NP problems" [Lucas].
3. In the process, the call $neighbour(state)$ generates a randomly chosen neighbour $state\_n$ of a given current state *state*.
4. The SA cooling schedule (crucial component) is defined by the call $temperature(i)$, which yields the temperature to use. The Julia code includes five different cooling schedules which can be used by plugging them into the $simulated\_annealing()$ function call.

I had previously worked only with matrices in Julia so it was fun to learn more about the language but maybe not as much fun to fix all the errors that cropped up. It was interesting to learn about how Julia doesn't have classes in the OOP sense and has mutable structs instead. I found them being primarily used in Optim.jl SA algorithm's source code.

While working on this task, I came across two interesting papers: one describing parametric cooling schedules (reference paper) and the other regarding how to find an optimal initial temperature for SA (reference paper).

III. References

1. https://github.com/JuliaNLSolvers/Optim.jl
2. https://dl.acm.org/doi/pdf/10.5555/3408352.3408509
3. https://www.mathworks.com/matlabcentral/answers/uploaded_files/14677/B:COAP.0000044187.23143.bd.pdf
4. https://github.com/recruit-communications/pyqubo/blob/master/notebooks/integer_partition.ipynb
5. https://www.mathworks.com/matlabcentral/answers/uploaded_files/14677/B:COAP.0000044187.23143.bd.pdf