Tutorial: Markov Chain Monte Carlo Methods for Various Optimization Problems*

*For ECE 506 project, Fall 2018

Mustafa S Salman
Dept. of Electrical & Computer Engineering
University of New Mexico
Albuquerque, NM, USA
esalman@unm.edu

***Abstract*—There are several well-known and tested methods for optimization of differentiable and deterministic problems, including but not limited to the gradient descent, BFGS and Newton's methods and their variants. On the other hand, Markov Chain Monte Carlo methods are useful for solution of non-differentiable and non-deterministic problems. The goal of this project is to study and compare the performance of Markov Chain Monte Carlo methods for various deterministic and non-deterministic optimization NP-hard problems.**

***Index Terms*—optimization, gradient descent, BFGS, Newton's method, Markov Chain Monte Carlo, MCMC**

I. Motivation

In many circumstances, mathematical models of physical systems becomes so complex that inverse methods cannot be used to compute the model parameters. Markov-chain Monte Carlo (MCMC) methods are simple numerical approaches appropriate for this type of situations. MCMC method can be used to determine an optimal parameter set for a model of arbitrary complexity by minimizing a cost function [6]. It does not require the estimation of the gradient of the goodness-of-fit function, thus has an advantage over gradient descent methods [7] in optimizing non-differentiable functions. There exists non-deterministic optimization problems with many local optima and complicated constraints, and a possible mix of continuous and discrete variables. MCMC is at the core of randomized optimization methods such as stochastic approximation, simulated annealing, evolutionary algorithms and cross-entropy methods which can be used to solve both random and deterministic problems [8]. Therefore I am interested in studying MCMC methods.

II. Expected Contribution

1) Implementing deterministic algorithm(s) (steepest descent, BFGS, Newton's method etc.) in Python
2) Implementing randomized optimization algorithm(s) (stochastic approximation, ~~simulated annealing~~, evolutionary algorithm, cross-entropy method etc.) in Python
3) Using randomized optimization algorithm(s) to solve deterministic problem(s) and compare the validation metrics with the deterministic algorithm(s)
4) Using randomized optimization algorithm(s) to solve NP-hard problem(s) and (if possible) compare with the deterministic algorithms

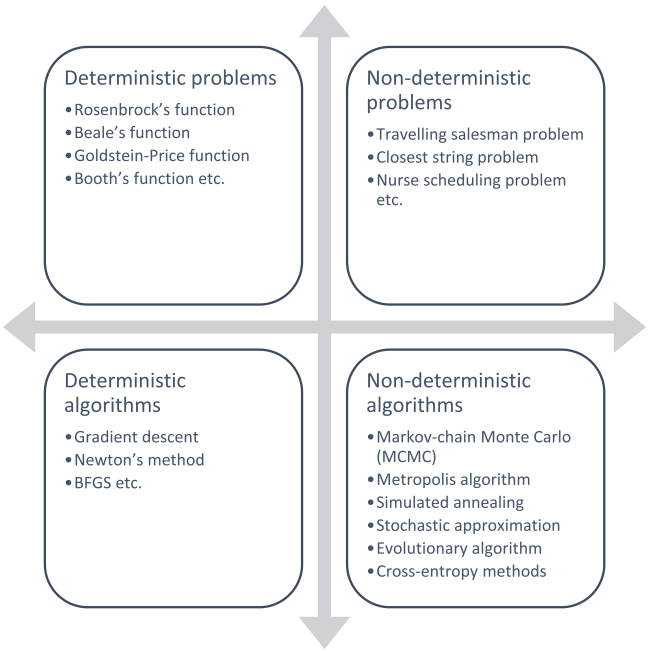


Fig. 1. Various types of optimization problems and algorithms

Fig. 1 shows a summary of the types of probelems and algorithms which may be covered in this project.

III. Code

I intend to use Python for this project. Fig. 2 shows a screenshot of the project template folder. Following is the description of the contents of the template folder:

.git contains Git versioning information.
bin contains libraries and packages required to run the project.

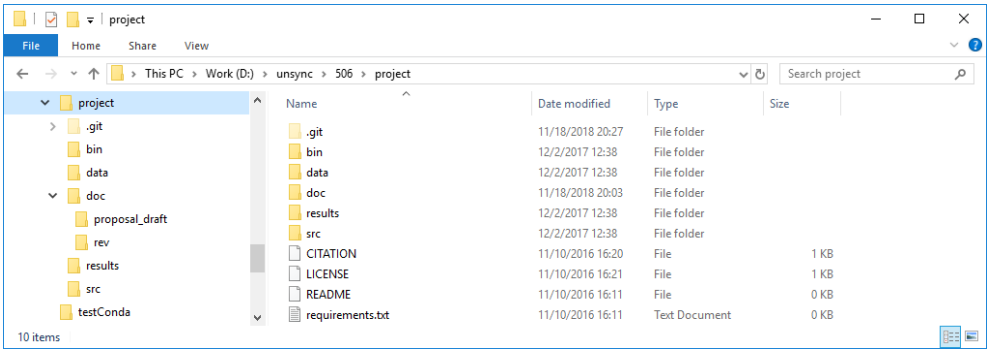


Fig. 2. Project template directory

data contains unprocessed data acquired for using in the project.

doc contains all the documentations (revision, project proposal etc.), including the LaTeXsources. I have chosen LaTeXover Microsoft Word which is difficult to maintain. Although Jupyter Notebook would be a great tool for maintaining the source code and documentation, it is difficult for version control and I also intend to write the project proposal and other documents in the IEEE format instead of in a Jupyter Notebook.

results contains processed data and any other output from running the source code, including figures.

src contains the source codes.

CITATION contains the information about how to cite this project.

README brief project README document.

LICENSE contains copyright and licensing information of the project.

req...s.txt required for automatic deployment.

IV. Data

For this project I have searched for problems/functions to optimize. Here I am listing some problems that I have found which might be worth investigating for the use of the algorithms.

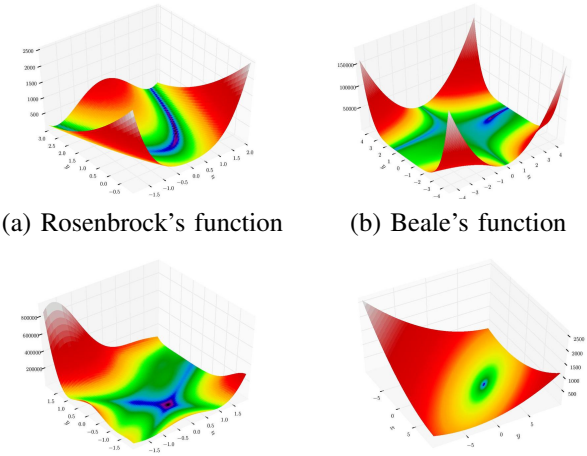
A. Differentiable/Deterministic Problems

This classes of functions will be obtained from Wikipedia [3]. Some examples are shown in Fig. 3

B. Non-differentiable/Non-deterministic Problems

Following are some examples of NP-hard problems which tend to be non-deterministic and may be solved by MCMC methods.

- **The travelling salesman problem (TSP)**: It asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization [2].



(a) Rosenbrock's function (b) Beale's function

(c) Goldstein-Price function (d) Booth's function

Fig. 3. Examples of test functions for optimization.

- **The closest string**: It is an NP-hard computational problem in theoretical computer science, which tries to find the geometrical center of a set of input strings. More formally, given n length-m strings $s1, s2, ..., sn$, the closest string problem seeks for a new length-m string s such that $d(s, si)k$ for all i, where d denotes the Hamming distance, and where k is as small as possible [4].
- **The nurse scheduling problem (NSP)**: Also called the nurse rostering problem (NRP), it is the operations research problem of finding an optimal way to assign nurses to shifts, typically with a set of hard constraints which all valid solutions must follow, and a set of soft constraints which define the relative quality of valid solutions [5].

V. Validation Metrics

For either class of problems, I will evaluate the following metrics:

- **Robustness**: Convergence for different random initializations- can be evaluated for both deterministic and non-deterministic problems.

- **Efficiency**: Required memory and number of function evaluations- can be evaluated for either type of problems.
- **Accuracy**: For deterministic problems as well as low-dimensional non-deterministic problems, it is feasible to determine the optimal solution and check the accuracy in terms of mean squared error of the proposed solution. However, when the non-deterministic problems increase in dimension, determining the optimal solution becomes hard (hence they are known as NP-hard problems)

REFERENCES

[1] A. Bock, E. Grant, J. Konemann, and L. Sanita, The School Bus Problem on Trees, p. 10. link

[2] Travelling salesman problem, Wikipedia. 16-Nov-2018. link

[3] Test functions for optimization, Wikipedia. 22-Oct-2018. link

[4] Closest string, Wikipedia. 09-Jun-2018. link

[5] Nurse scheduling problem, Wikipedia. 21-Aug-2018. link

[6] M. Herman, Simulated Annealing & the Metropolis Algorithm: A Parameter Search Method for Models of Arbitrary Complexity, p. 14. link

[7] S. Towers, Markov Chain Monte Carlo parameter optimization method — Polymatheia. link

[8] D. P. Kroese, T. Taimre, and Z. I. Botev, Handbook of Monte Carlo Methods. John Wiley & Sons, 2013. link