Locally-informed proposals in Metropolis-Hastings algorithm with applications

Bartosz Chmiela

Supervisor: dr. hab Paweł Lorek

University of Wrocław

08.07.2022

Contents

1 Markov chains

2 Markov chain Monte Carlo methods

3 Traveling salesman problem

4 Approximate solutions

5 Results

6 Conclusions

Markov chain

Definition

A Markov chain is a stochastic process describing a sequence on some countable state space S in which the probability of each event depends only on the state of previous event.

Definition

Such a chain is uniquely defined by a stochastic transition matrix $\mathbf{P}$ and a initial distribution μ.

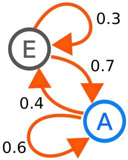


Figure: A diagram representing a two-state Markov chain, source: wiki.

Properties

Definition (Irreducibility)

A Markov chain is irreducible if there is a possibility to reach every state from every state.

Definition (Aperiodicity)

A state is aperiodic if there is no possibility of coming back after some period. A Markov chain is aperiodic when is irreducible and a state is aperiodic.

Ergodic chains

Definition (Stationarity)

A probability distribution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ is called stationary if it satisfies

$$\pi_j = \sum_{i \in S} \pi_i p_{ij},$$

Definition (Ergodicity)

A Markov chain is ergodic when it is irreducible and aperiodic.

Theorem

Let $\{X_k\}$ *be an ergodic Markov chain with a transition matrix* $\mathbf{P}$, *then:*

$$\lim_{k \to \infty} p_{ij}(k) = \pi_j.$$

Metropolis-Hastings algorithm

Construct a MC, which has a stationary distribution $\boldsymbol{\pi}$ $(\pi_i > 0)$. Assume that $\mathbf{Q}$ is stochastic matrix which corresponds to an irreducible and aperiodic Markov chain and $\mathbf{Q}_{i,j} > 0 \iff \mathbf{Q}_{j,i} > 0$. Let us consider a matrix defined as:

$$\mathbf{P}_{i,j} = \begin{cases} \mathbf{Q}_{i,j} \min\left(1, \frac{\pi_j \mathbf{Q}_{j,i}}{\pi_i \mathbf{Q}_{i,j}}\right) & \text{if } i \neq j, \\ 1 - \sum_{j \in S \setminus \{i\}} \mathbf{P}_{i,j} & \text{if } i = j. \end{cases} \tag{1}$$

Theorem

A matrix defined in (1) is stochastic, irreducible, aperiodic and has a stationary distribution $\boldsymbol{\pi}$.

Metropolis-Hastings algorithm

Algorithm Metropolis-Hastings algorithm

```
Choose a state $i \in S$.
$X_0 \leftarrow i$
for $k = 0, 1, \ldots$ do
    Sample $j \sim \mathbf{Q}_i = (\mathbf{Q}_{i,1}, \mathbf{Q}_{i,2}, \ldots, \mathbf{Q}_{i,N})$.
    Sample $U \sim Unif(0, 1)$.
    if $U \leq \min\left(1, \frac{\pi_j \mathbf{Q}_{j,i}}{\pi_i \mathbf{Q}_{i,j}}\right)$ then
        $X_{k+1} \leftarrow j$
    else
        $X_{k+1} \leftarrow X_k$
    end if
end for
```

Traveling salesman problem

Definition (TSP)

Given a list of cities and the distances between each pair of cities, what is the shortest possible tour that visits each city exactly once and returns to the origin city?

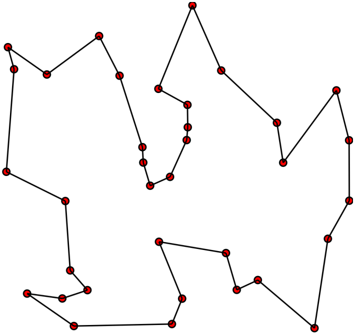


Figure: Traveling salesman problem, source: wiki.

Traveling salesman problem

Definition (Traveling salesman problem)

Given an undirected weighted graph $G = (V, E)$, $|V| = n$ find a permutation $\sigma_{\min}$ of vertices such that

$$\sigma_{\min} = \arg\min_{\sigma \in S_n} \left(\sum_{i=1}^{n-1} w_{\sigma(i),\sigma(i+1)} + w_{\sigma(n),\sigma(1)} \right),$$

where S_n is a set of all permutations of vertices and $w_{i,j}$ is distance (weight) between cities i and j.

Softmax

Definition

For a given vector $\mathbf{x} = (x_1, x_2, \ldots, x_d)^T \in \mathbb{R}^d$ a softmax function $s : \mathbb{R}^d \to [0,1]^d$ is defined as

$$s(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}},$$
$$s(\mathbf{x}) = (s(\mathbf{x})_1, s(\mathbf{x})_2, \ldots, s(\mathbf{x})_d)\,.$$

$$\sigma_{\min} = \arg\min_{\sigma \in S_n} (w_\sigma) = \arg\max_{\sigma \in S_n} \frac{e^{-w_\sigma}}{\sum_{\sigma' \in S_n} e^{-w_{\sigma'}}},$$

Metropolis-Hastings algorithm

$$\pi_\sigma = \frac{e^{-w_\sigma}}{\sum_{\sigma' \in S_n} e^{-w_{\sigma'}}}$$

$$\frac{\pi_{\sigma'} \mathbf{Q}_{\sigma',\sigma}}{\pi_\sigma \mathbf{Q}_{\sigma,\sigma'}} = e^{-(w_{\sigma'} - w_\sigma)} \cdot \frac{\mathbf{Q}_{\sigma',\sigma}}{\mathbf{Q}_{\sigma,\sigma'}}.$$

$$\log\left(\frac{\pi_{\sigma'} \mathbf{Q}_{\sigma',\sigma}}{\pi_\sigma \mathbf{Q}_{\sigma,\sigma'}}\right) = -(w_{\sigma'} - w_\sigma) + \log(\mathbf{Q}_{\sigma',\sigma}) - \log(\mathbf{Q}_{\sigma,\sigma'}).$$

Neighbours

Definition

A neighbour σ' of a permutation σ is a permutation, that for some k, l it satisfies $\sigma'(k) = \sigma(l)$, $\sigma'(l) = \sigma(k)$ and $\sigma'(i) = \sigma(i)$ for the rest of indices.

These neighbours are the original tour with two swaped indices. This let us consider a smaller space – there are $\binom{n}{2} = \frac{n(n-1)}{2} \approx n^2$ neighbours if the number of vertices is n.

Random neighbour (RN)

Sample neighbours uniformly. It is equivalent to choosing random indices to swap.

$$\mathbf{Q}_{\sigma,\sigma'} = \begin{cases} \frac{1}{|S_\sigma|} & \text{if } \sigma' \in S_\sigma, \\ 0 & \text{if } \sigma' \notin S_\sigma. \end{cases}$$

where S_σ is a set of all possible neighbours of σ.

Random candidates (RN)

When given a tour σ and its neighbour σ' they differ only on those edges where swap is happening, let us say k, l. So for this situation we have tours:

$$\begin{aligned}\sigma = (\dots, \sigma(k-1), \sigma(k), \sigma(k+1), \dots, \\ \dots, \sigma(l-1), \sigma(l), \sigma(l+1), \dots) \\ \sigma' = (\dots, \sigma(k-1), \sigma(l), \sigma(k+1), \dots, \\ \dots, \sigma(l-1), \sigma(k), \sigma(l+1), \dots)\end{aligned}$$

To compute difference $w_{\sigma'}$ we need to remove weights $w_{\sigma(k-1),\sigma(k)}$, $w_{\sigma(k),\sigma(k+1)}$, $w_{\sigma(l-1),\sigma(l)}$, $w_{\sigma(l),\sigma(l+1)}$ and add $w_{\sigma(k-1),\sigma(l)}, w_{\sigma(l),\sigma(k+1)}, w_{\sigma(l-1),\sigma(k)}, w_{\sigma(k),\sigma(l+1)}$.

Locally-informed proposals (LIP)

The idea is to balance the increase in the probability of neighbour with decrease of reverse probability, such that it will be easy to compute.

$$\mathbf{Q}_{\sigma,\sigma'} \propto e^{\frac{-(w_{\sigma'}-w_{\sigma})}{\tau}}.$$

The distribution is chosen in such a way, so that we can easily group up the terms in acceptance criterion:

$$\frac{\pi_{\sigma'}\mathbf{Q}_{\sigma',\sigma}}{\pi_{\sigma}\mathbf{Q}_{\sigma,\sigma'}} = e^{-(w_{\sigma'}-w_{\sigma})} \cdot \frac{e^{\frac{-(w_{\sigma}-w_{\sigma'})}{\tau}}}{e^{\frac{-(w_{\sigma'}-w_{\sigma})}{\tau}}} \cdot \frac{C_{\sigma'}}{C_{\sigma}} = e^{\left(-(w_{\sigma'}-w_{\sigma})\left(1-\frac{2}{\tau}\right)\right)} \cdot \frac{C_{\sigma'}}{C_{\sigma}}$$

Example

Let us set $k = 2$ and $l = 7$, then the set of indices to consider is $\{1, 2, 3, 6, 7, 8\}$ and the permutations:

$$\sigma = (1, \mathbf{2}, 3, 4, 5, 6, \mathbf{7}, 8, 9, \ldots)$$
$$\sigma' = (1, \mathbf{7}, 3, 4, 5, 6, \mathbf{2}, 8, 9, \ldots).$$

Example

	2	3	4	5	6	7	8	9	10
1	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)	(1,9)	···
2		(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)	(2,9)	···
3			(3,4)	(3,5)	(3,6)	(3,7)	(3,8)	(3,9)	···
4				(4,5)	(4,6)	(4,7)	(4,8)	(4,9)	···
5					(5,6)	(5,7)	(5,8)	(5,9)	···
6						(6,7)	(6,8)	(6,9)	···
7							(7,8)	(7,9)	···
8								(8,9)	···
9									⋱

Table: Neighbour representation.

Simulated annealing

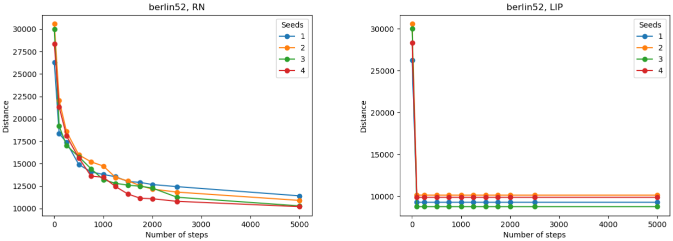
The idea is to describe a probability of state using a cooling parameter t_k such that it reminds the cooling of a metal and may change with each step:

$$\pi_i = \frac{e^{\frac{-E_i}{t_k}}}{C},$$

where C is normalizing constant.The quotient of probabilities then is:

$$\frac{\pi_j}{\pi_i} = e^{\frac{E_i - E_j}{t_k}}$$

Initial condition

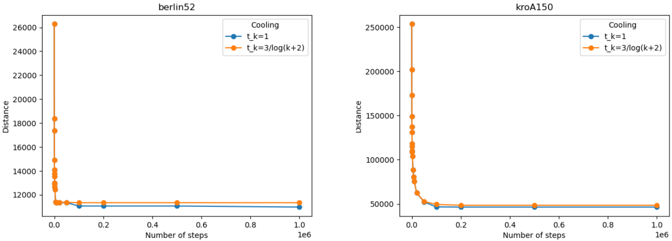


(a) Random neighbours.

(b) Locally-informed proposals.

Figure: Different initial states for *berlin52*.

Simulated annealing

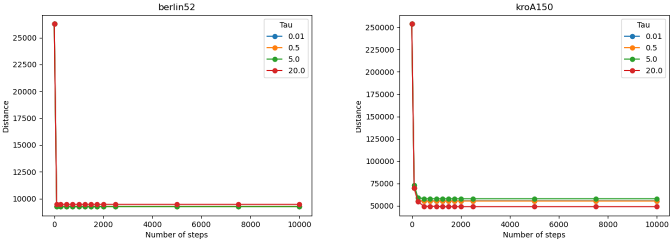


(a) berlin52.

(b) kroA150.

Figure: Different cooling parameters for *berlin52* and *kroA150*.

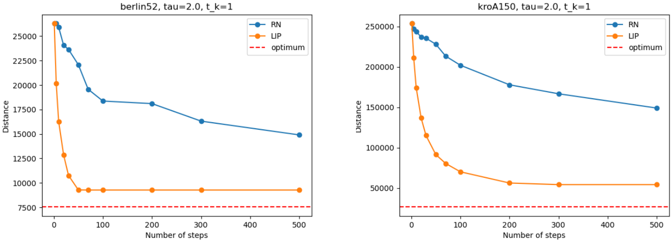
Temperature



(a) berlin52. (b) kroA150.

Figure: Different temperature parameters for *berlin52* and *kroA150*.

Algorithms comparison

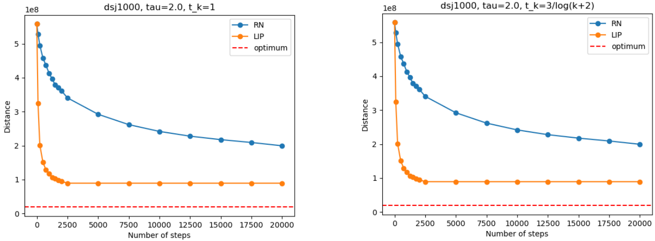


(a) berlin52, $t_k = 1$.

(b) kroA150, $t_k = 1$.

Figure: Comparing RN and LIP for *berlin52* and *kroA150* with low number of iterations.

Algorithms comparison



(a) dsj1000, $t_k = 1$.

(b) dsj1000, $t_k = \frac{3}{\log(k+2)}$.

Figure: Comparing RN and LIP for *dsj1000* with different cooling parameters.

Algorithms comparison

	RN			LIP		
dataset	dist.	time $[s]$	ratio	dist.	time $[s]$	ratio
berlin52	11344	1.32	1.50	9276	116	1.23
kroA150	62467	1.24	2.36	53988	454	2.04
att532	165445	1.38	5.98	78150	2462	2.82
dsj1000	199343433	1.5	10.68	89105687	6439	4.78

Table: Methods comparison, $t_k = 1$, 20000-th step.

Algorithms comparison

	RN			LIP		
dataset	step	distance	time [s]	step	distance	time [s]
berlin52	20000	11344	1.32	27	11198	0.22
kroA150	20000	62467	1.24	139	62459	3.64
att532	20000	165445	1.38	199	165247	24.65
dsj1000	20000	199343433	1.5	256	199338468	80.54

Table: Amount of time and number of steps required for LIP algorithm to reach the result of RN after 20000 steps, $t_k = 1$.

Conclusions

- LIP algorithm is decreasing the distance quicker than RN,
- LIP converges to some value, which is always smaller than the RN,
- LIP reaches better results in feasible time,
- Only first hundreds iterations matter,
- LIP algorithm is considerably slower for a longer run.

Improvements

- Use of interning in *Python3*,
- use of concurrency,
- sampling from $\mathbf{Q}_\sigma$ using MCMC methods,
- better tuning of temperature and cooling parameters.

Thank you for your attention.