Locally-informed proposals in Metropolis-Hastings algorithm with applications

Chmiela Bartosz

University of Wrocław

08.07.2022

Contents

Markov chain

Definition (State space)

A state space of a Markov chain is a countable set S.

Definition (Index set)

An index set of a Markov chain is a countable set T.

Markov chain

Definition

A Markov chain is a sequence of random variables $\{X_k\}_{k\in T}$ defined on a common probability space $(\Omega, \mathcal{F}, P)$, that take values in S, such that it satisfies Markov property:

$$P(X_{k+m} = j | X_k = i, X_{l_{p-1}} = i_{l_{p-1}}, \ldots, X_{l_1} = i_1) = \\ = P(X_{k+m} = j | X_k = i),$$

for all indices $l_1 < \ldots < l_{p-1} < k < k+m$, $1 \leq p \leq k$, all states $j, i, i_{p-1}, i_{p-2}, \ldots, i_0 \in S$ and $m \geq 1$.

Properties

Definition (Irreducibility)

A Markov chain with transition matrix $\mathbf{P}$ is called irreducible if and only if for every pair of states i and j there exists a positive probability of transition between them.

Definition (Periodicity)

Let d_i be a greatest common divisor of those k such that $\mathbf{P}_{i,i}(k) > 0$. If $d_i > 1$ then state i is periodic. If $d_i = 1$ then state i is aperiodic.

Properties

Definition (Stationarity)

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

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

or equivalently in vector form:

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}.$$

This equation is often described as the balance equation.

Definition (Ergodicity)

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

Ergodic chains

Theorem

Let $\{X_k\}$ *be a ergodic Markov chain, 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 is irreducible, aperiodic 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

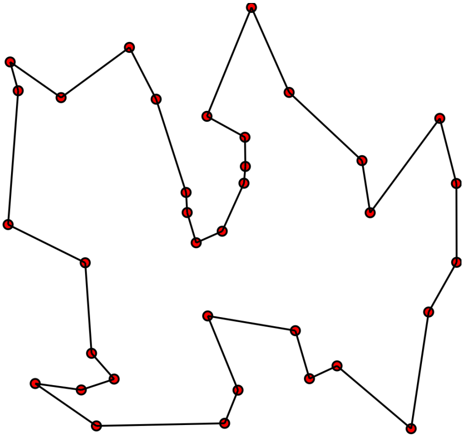


Figure: Traveling salesman problem, source: wiki.

Traveling salesman problem

Definition

A undirected graph G is a pair (V, E), where V is a set of vertices and E is a set of edges, which is a subset of all unordered pairs of vertices.

Definition

A tour is a Hamiltonian cycle and we identify it with a permutation of vertices.

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 state 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

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

$$s(\mathbf{x}) = (s(\mathbf{x})_1, s(\mathbf{x})_2, \ldots, s(\mathbf{x})_d)\,.$$

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

Metropolis-Hastings algorithm

$$\frac{\pi_j \mathbf{Q}_{j,i}}{\pi_i \mathbf{Q}_{i,j}} = e^{-(w_j - w_i)} \cdot \frac{\mathbf{Q}_{j,i}}{\mathbf{Q}_{i,j}}.$$

$$\log\left(\frac{\pi_j \mathbf{Q}_{j,i}}{\pi_i \mathbf{Q}_{i,j}}\right) = -(w_j - w_i) + \log(\mathbf{Q}_{j,i}) - \log(\mathbf{Q}_{i,j}).$$

Candidates

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 candidates (RN)

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

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:

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

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 $w_{\sigma(k-1),\sigma(l)}, w_{\sigma(l),\sigma(k+1)}, w_{\sigma(l-1),\sigma(k)}, w_{\sigma(k),\sigma(l+1)}$.

Random candidates (RN)

Algorithm Random neighbours algorithm

Choose a tour $\sigma \in S_n$.
$X_0 \leftarrow \sigma$
Compute weight w_σ.
for $i = 0, 1, \ldots$ **do**
    Sample $k, l \sim Unif\, \{1, 2, \ldots, n\}$ without replacement.
    Sample $U \sim Unif(0, 1)$.
    $w_{\sigma'} \leftarrow w_\sigma - (w_{\sigma(k-1)+\sigma(k)} + w_{\sigma(k)+\sigma(k+1)}, w_{\sigma(l-1)+\sigma(l)} + w_{\sigma(l),\sigma(l+1)})$
    $w_{\sigma'} \leftarrow w'_\sigma + (w_{\sigma(k-1)+\sigma(l)} + w_{\sigma(l)+\sigma(k+1)} + w_{\sigma(l-1)+\sigma(k)} + w_{\sigma(k)+\sigma(l+1)})$
    if $\log(U) \leq \min\left(0, -(w_{\sigma'} - w_\sigma)\right)$ **then**
        $X_{i+1} \leftarrow X_i$
        $X_{i+1}(k), X_{i+1}(l) \leftarrow X_{i+1}(l), X_{i+1}(k)$
        $w_\sigma \leftarrow w_{\sigma'}$
    else
        $X_{i+1} \leftarrow X_i$
    end if
end for

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.

$$\mathbf{Q}_{i,j} \propto e^{\frac{-(w_j - w_i)}{\tau}}.$$

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

$$\frac{\pi_j \mathbf{Q}_{j,i}}{\pi_i \mathbf{Q}_{i,j}} = e^{-(w_j - w_i)} \cdot \frac{e^{\frac{-(w_i - w_j)}{\tau}}}{e^{\frac{-(w_j - w_i)}{\tau}}} \cdot \frac{C_j}{C_i} = e^{\left(-(w_j - w_i)\left(1 - \frac{2}{\tau}\right)\right)} \cdot \frac{C_j}{C_i},$$

Locally-informed proposals (LIP)

Choose tour σ and its neighbour σ' that is connected with swapping indices k and l.

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

Locally-informed proposals (LIP)

The neighbours $\sigma_{r,s}$, $\sigma'_{r,s}$ of σ and σ' respectively look like:

$$\begin{aligned}
\sigma_{r,s} =& (\ldots, \sigma(r-1), \sigma(s), \sigma(r+1), \ldots, \\
& \ldots, \sigma(s-1), \sigma(r), \sigma(s+1), \ldots, \\
& \ldots, \sigma(k-1), \sigma(k), \sigma(k+1), \ldots, \\
& \ldots, \sigma(l-1), \sigma(l), \sigma(l+1), \ldots) \\
\sigma'_{r,s} =& (\ldots, \sigma(r-1), \sigma(s), \sigma(r+1), \ldots \\
& \ldots, \sigma(s-1), \sigma(r), \sigma(s+1), \ldots, \\
& \ldots \sigma(k-1), \sigma(l), \sigma(k+1), \ldots, \\
& \ldots, \sigma(l-1), \sigma(k), \sigma(l+1), \ldots)
\end{aligned}$$

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:

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

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.

Example

Let us set $r = 3$ and $s = 9$, which means that we are looking for weight differences of neighbours obtained by swapping 3 and 9. So the neighbours have form:

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

$$w_{\sigma_{3,9}} - w_{\sigma'_{3,9}} = (w_{1,2} + w_{2,9} + w_{6,7} + w_{7,8}) - (w_{1,7} + w_{7,9} + w_{6,2} + w_{2,8}).$$

We can generalize that equation for any $p \notin \{1, 2, 3, 6, 7, 8\}$:

$$w_{\sigma_{3,p}} - w_{\sigma'_{3,p}} = f_3(p) = C + w_{2,p} - w_{7,p}.$$

Locally-informed proposals (LIP)

Algorithm Locally-informed proposals algorithm

Choose a tour $\sigma \in S_n$.
$X_0 \leftarrow \sigma$
Compute weight w_σ.
Compute all neighbour weight differences $\boldsymbol{d}_\sigma$.
$s(\boldsymbol{d}_\sigma) = \mathrm{softmax}(\boldsymbol{d}_\sigma)$
for $i = 0, 1, \ldots$ **do**
  Sample $\sigma' \sim s(\boldsymbol{d}_\sigma)$.
  Find k, l connected with swapping.
  $w_{\sigma'} \leftarrow w_\sigma + \boldsymbol{d}_\sigma[(k,l)]$
  $\boldsymbol{d}_{\sigma'} \leftarrow \boldsymbol{d}_\sigma$
  $\boldsymbol{d}_{\sigma'} \leftarrow$ *update_differences*$(\boldsymbol{d}_{\sigma'})$
  $s(\boldsymbol{d}_{\sigma'}) = \mathrm{softmax}(\boldsymbol{d}_{\sigma'})$
  Sample $U \sim Unif(0,1)$.
  if $\log(U) \leq \min\left(0, -(w_{\sigma'} - w_\sigma) + \log(s(\boldsymbol{d}_{\sigma'})[(k,l)]) - \log(s(\boldsymbol{d}_\sigma)[(k,l)])\right)$ **then**
    $X_{i+1} \leftarrow X_i$
    $X_{i+1}(k), X_{i+1}(l) \leftarrow X_{i+1}(l), X_{i+1}(k)$
    $w_\sigma \leftarrow w_{\sigma'}$
    $\boldsymbol{d}_\sigma \leftarrow \boldsymbol{d}_{\sigma'}$
  else
    $X_{i+1} \leftarrow X_i$
  end if
end for

Locally-informed proposals (LIP)

Algorithm update_differences

Require: $\boldsymbol{d}_{\sigma'}$
Ensure: $\boldsymbol{d}_{\sigma'}$

for $r = k-1, k, k+1, l-1, l, l+1$ **do**
    for $s = r+1, \ldots, n$ **do**
        $\boldsymbol{d}_{\sigma'}[(r,s)] \leftarrow \textit{get\_difference}(\boldsymbol{d}_{\sigma'}[(r,s)])$
    end for
end for
for $s = k-1, k, k+1, l-1, l, l+1$ **do**
    for $r = 1, 2, \ldots l$ **do**
        $\boldsymbol{d}_{\sigma'}[(r,s)] \leftarrow \textit{get\_difference}(\boldsymbol{d}_{\sigma'}[(r,s)])$
    end for
end for

Simulated annealing

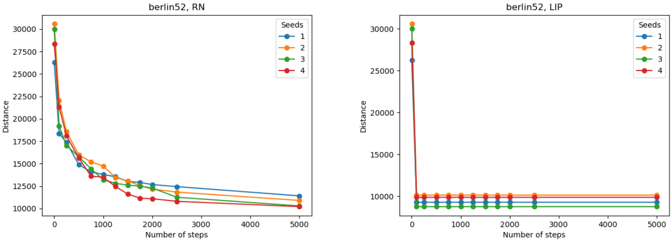
The idea is to describe a probability of state using a cooling parameter t_k such that it reminds the cooling of a metal and may change with each step:

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

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

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

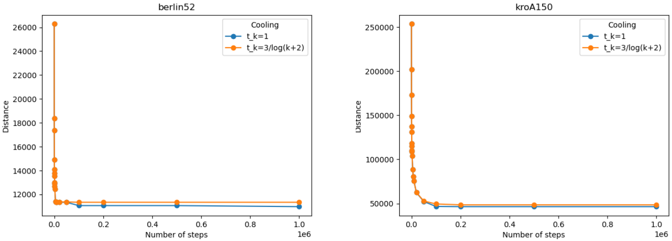
Initial condition



(a) Random neighbours. (b) Locally-informed proposals.

Figure: Different initial states for *berlin52*.

Simulated annealing

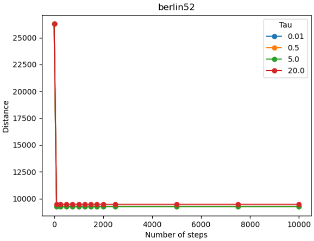


(a) berlin52.

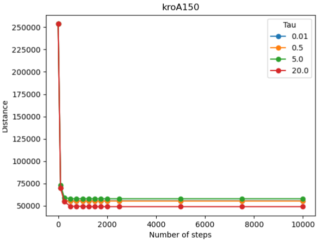
(b) kroA150.

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

Temperature



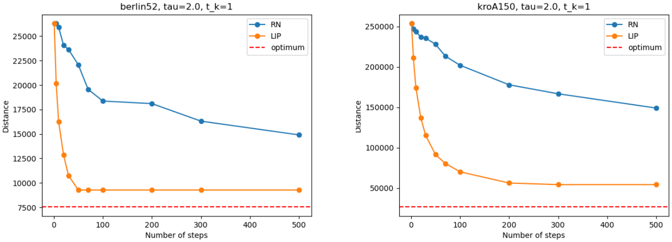
(a) berlin52.



(b) kroA150.

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

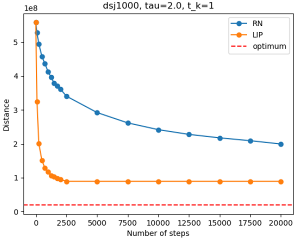
Algorithms comparison



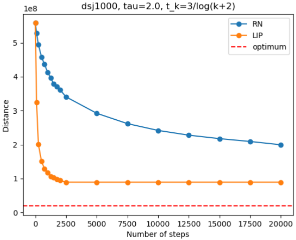
(a) berlin52, $t_k = 1$. (b) kroA150, $t_k = 1$.

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

Algorithms comparison



(a) $t_k = 1.$



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

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 *Pyton3*,
- use of concurrency,
- sampling from $\mathbf{Q}_i$ using MCMC methods,
- better tuning of temperature and cooling parameters.

Thank you for your attention.