BAT.jl

Multi-proposal Monte Carlo

Current status and plan for future development

Currently implemented algorithms:

- Tjelmeland's "Multi-Proposal MCMC"

  based on: *Using All Metropolis-Hastings Proposals to Estimate Mean Values [Tjelmeland, (2004)]*

- Liu's "Multiple-Try Metropolis"

  - based on: *The Multiple-TryMethod and Local Optimization in Metropolis Sampling [Liu, Liang, Wong, 2000]*

Multi-Proposal MCMC

- Large wealth of samples for the final estimator:
  - propose multiple samples
  - keep all samples with specific weights
  - rejection probability for next sample in chain is approx. 0
- Parallelization between steps
- Shorter burn-in phase
- Highly dependent on number of proposed samples and acceptance ratio for tuning
- Tricky selection of proposal distribution
- Ad-hoc tuning for different distributions

Multi-Try Metropolis (MTM)

Let $\boldsymbol{x}$ be the current state, $\boldsymbol{T(x, y)}$ the proposal distribution that propose a new sample $\boldsymbol{y}$ from the current one, and let $\boldsymbol{\pi(.)}$ be the target distribution.

We define the following quantity:

$$w(\boldsymbol{y}, \boldsymbol{x}) = \pi(\boldsymbol{y})\, T(\boldsymbol{y}, \boldsymbol{x})\, \lambda(\boldsymbol{y}, \boldsymbol{x})$$

Where $\lambda(\boldsymbol{x}, \boldsymbol{y})$ is a nonnegative symmetric function that can be chosen by the user. The only requirement is that $\lambda(\boldsymbol{x}, \boldsymbol{y}) > 0$ whenever $T(\boldsymbol{x}, \boldsymbol{y}) > 0$.

Multiple-try Metropolis

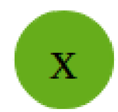
1. Draw k iid trial proposals, $\mathbf{y}_1, \ldots, \mathbf{y}_k$, from $T(\mathbf{x}, \cdot)$. Compute $w(\mathbf{y}_j, \mathbf{x})$ for $j = 1, \ldots, k$.

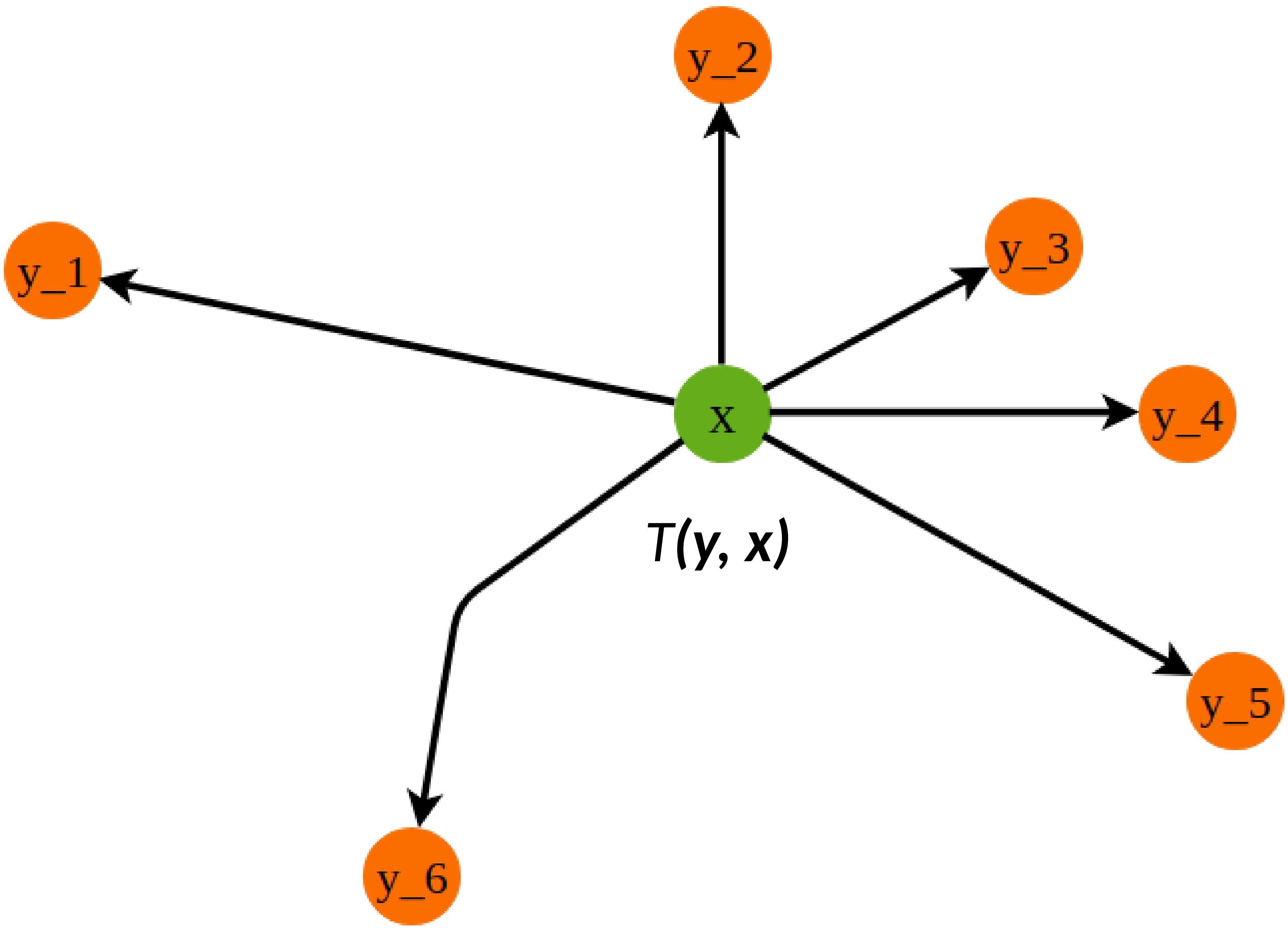
2. Select $\mathbf{Y} = \mathbf{y}$ among the trial set $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ with probability proportional to $w(\mathbf{y}_j, \mathbf{x}), j = 1, \ldots, k$. Then draw $\mathbf{x}_1^*, \ldots, \mathbf{x}_{k-1}^*$ from the distribution $T(\mathbf{y}, \cdot)$, and let $\mathbf{x}_k^* = \mathbf{x}$.

3. Accept $\mathbf{y}$ with probability

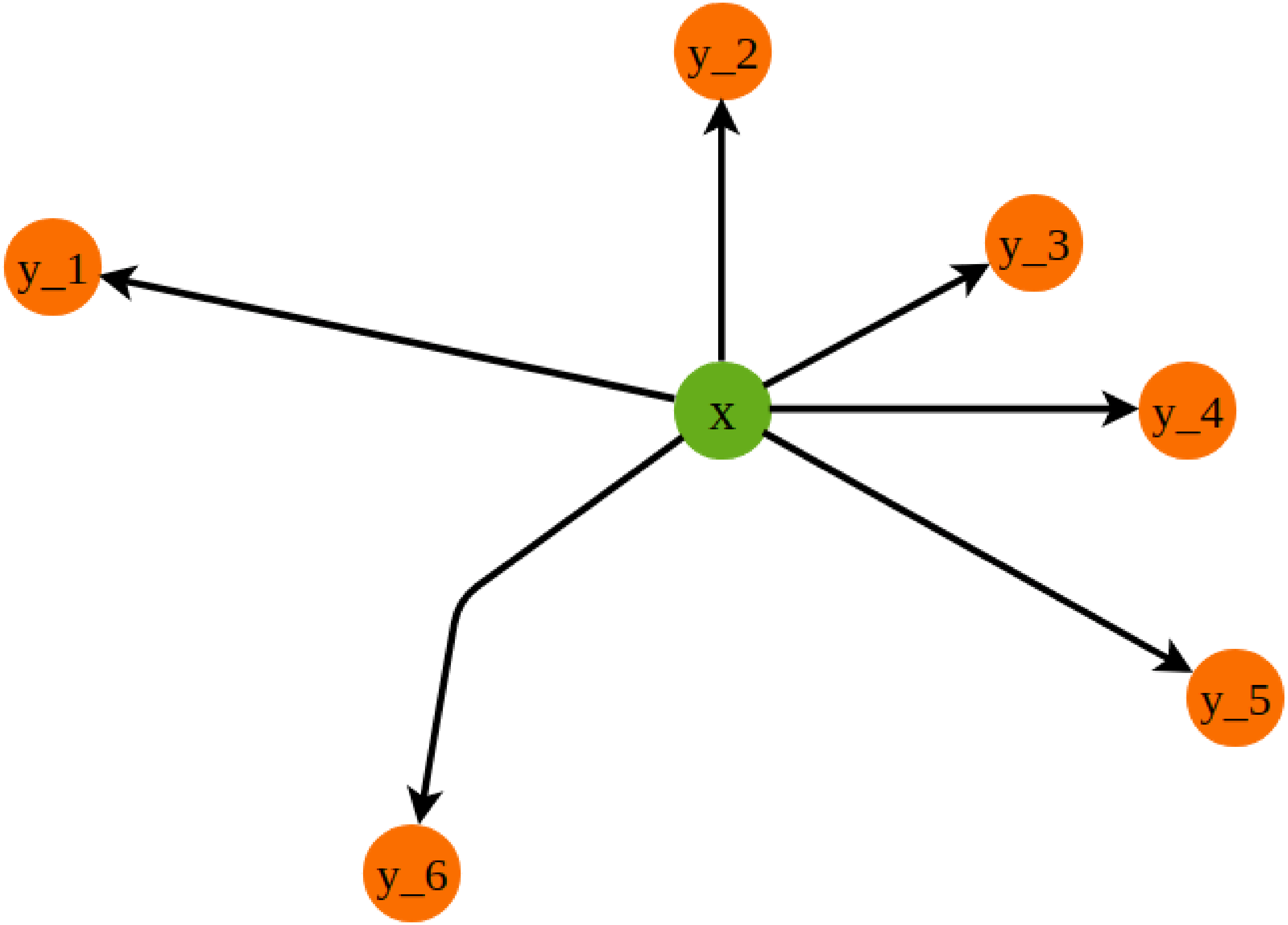
$$r_g = \min\left\{1, \frac{w(\mathbf{y}_1, \mathbf{x}) + \cdots + w(\mathbf{y}_k, \mathbf{x})}{w(\mathbf{x}_1^*, \mathbf{y}) + \cdots + w(\mathbf{x}_k^*, \mathbf{y})}\right\} \qquad (3)$$

and reject it with probability $1 - r_g$. The quantity r_g is called the *generalized M–H ratio.*

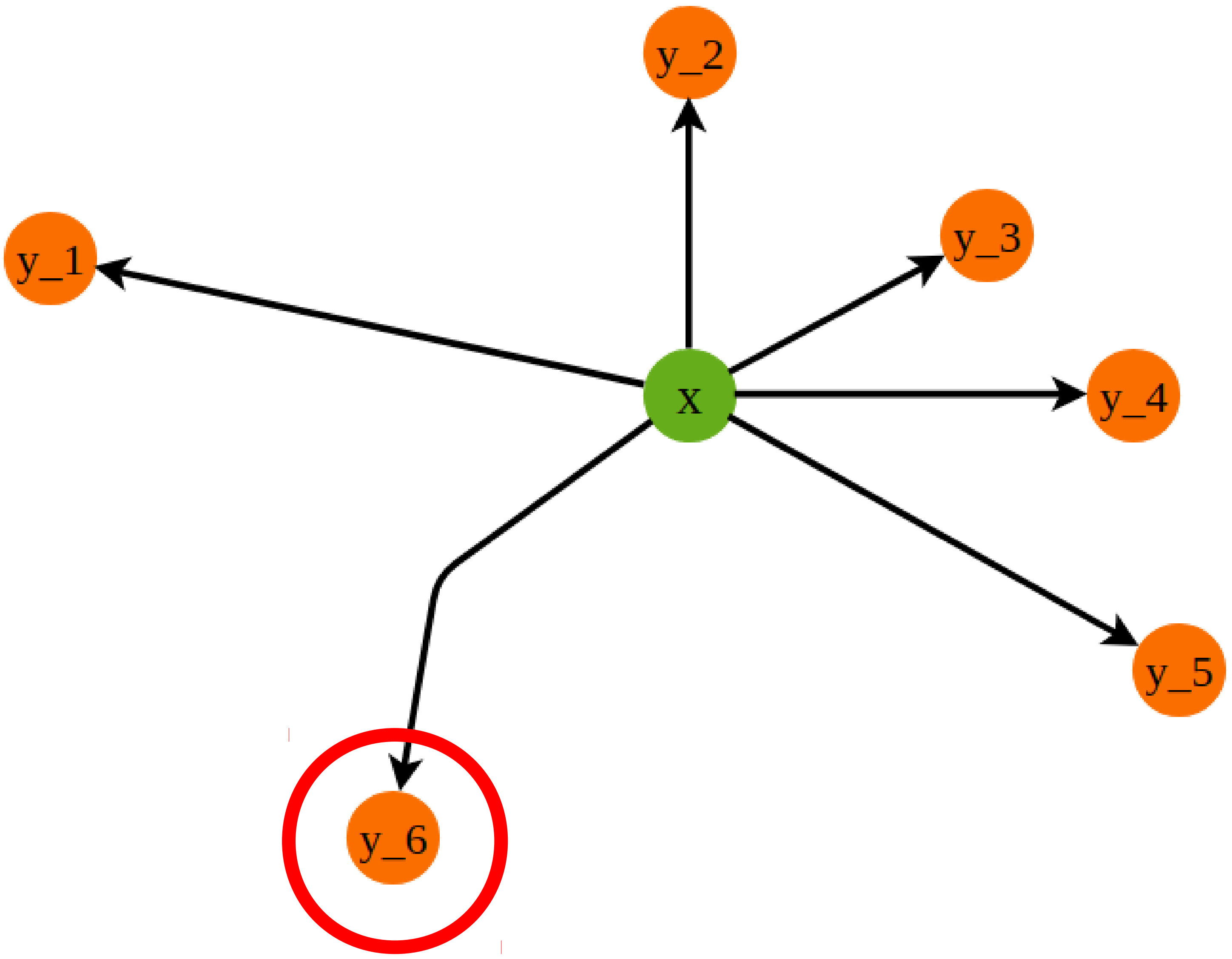

x

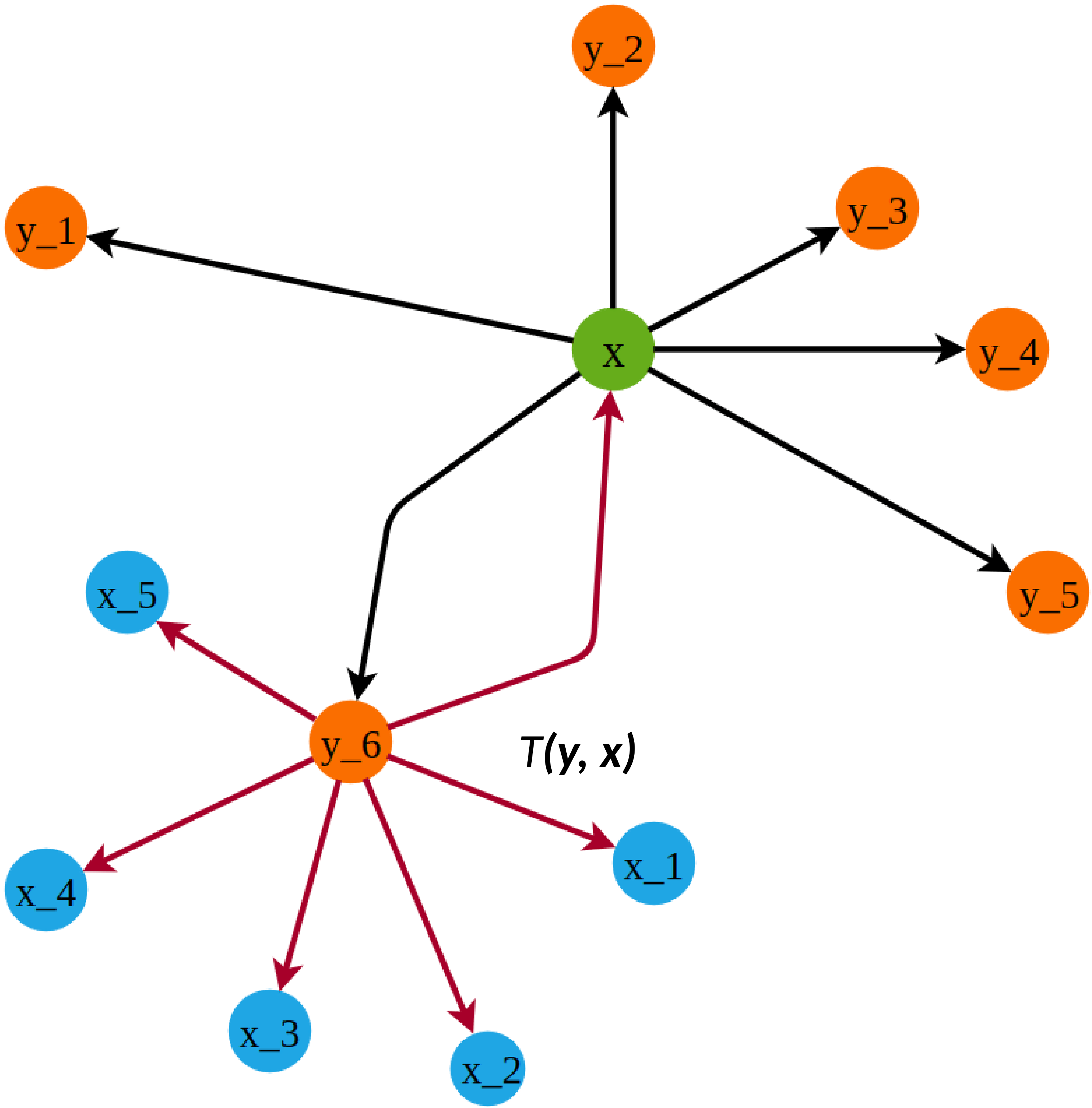

y_2
y_1
y_3
x
y_4
T(y, x)
y_5
y_6

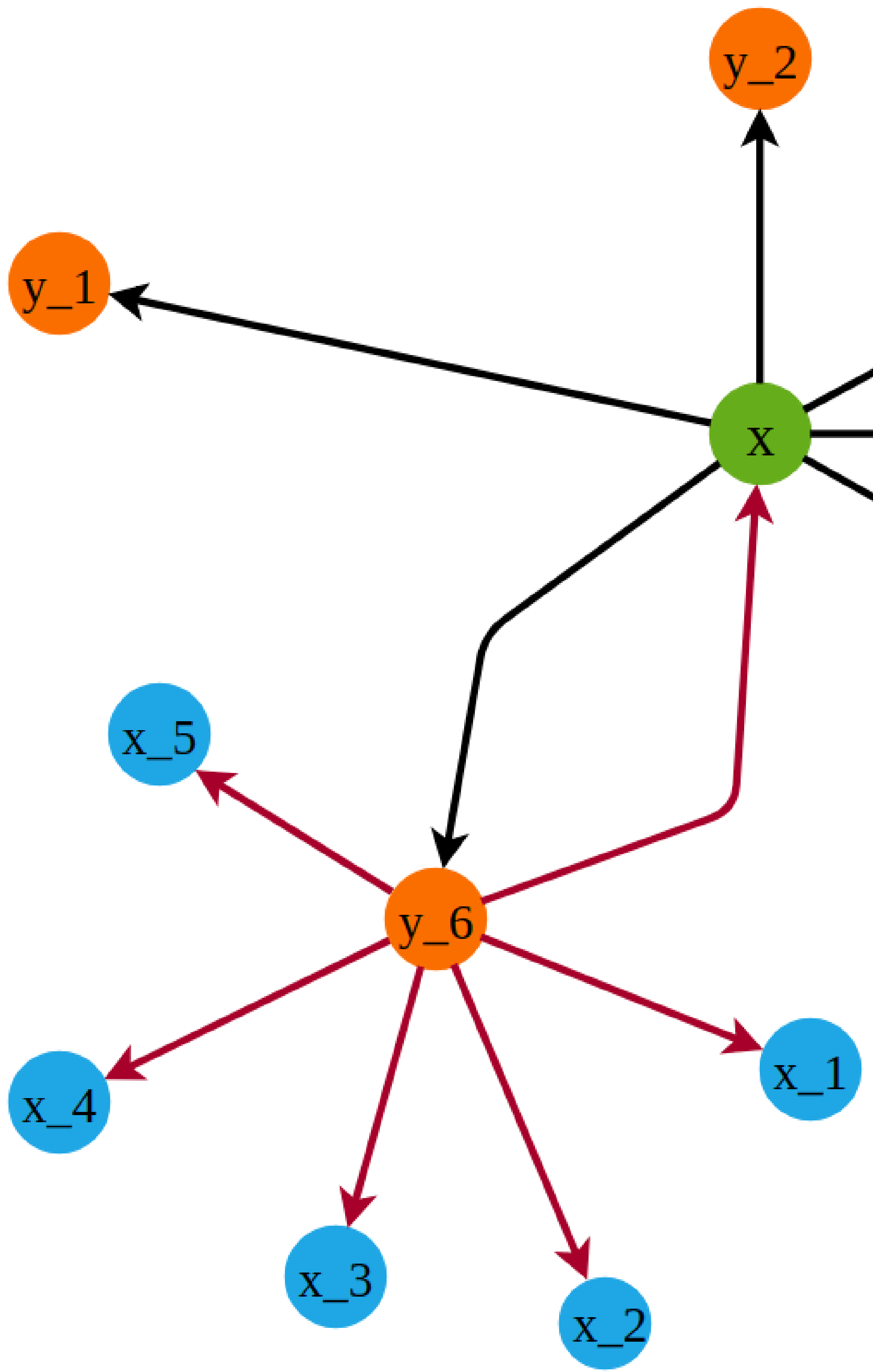
$w(\mathbf{y}, \mathbf{x}) = \pi(\mathbf{y})\ T(\mathbf{y}, \mathbf{x})\ \lambda(\mathbf{y}, \mathbf{x})$



$$w(\mathbf{y}, \mathbf{x}) = \pi(\mathbf{y})\, T(\mathbf{y}, \mathbf{x})\, \lambda(\mathbf{y}, \mathbf{x})$$




y_2
y_1
y_3
x
y_4
y_5
x_5
y_6
T(y, x)
x_1
x_4
x_3
x_2



$$w(\mathbf{y}, \mathbf{x}) = \pi(\mathbf{y})\, T(\mathbf{y}, \mathbf{x})\, \lambda(\mathbf{y}, \mathbf{x})$$

MTM

Possible choices of ***λ(y, x):***

- ***MTM1:***

$$\lambda(\mathbf{x}, \mathbf{y}) \equiv 1$$

- ***MTM2:***

$$\lambda(\mathbf{x}, \mathbf{y}) = \left(\frac{T(\mathbf{x}, \mathbf{y}) + T(\mathbf{y}, \mathbf{x})}{2} \right)^{-1} \longrightarrow \min \left\{ 1, \frac{\pi(\mathbf{y}_1) + \cdots + \pi(\mathbf{y}_k)}{\pi(\mathbf{x}_1^*) + \cdots + \pi(\mathbf{x}_k^*)} \right\}$$

- ***MTM3:***

$$\lambda(\mathbf{x}, \mathbf{y}) = \{T(\mathbf{x}, \mathbf{y}) T(\mathbf{y}, \mathbf{x})\}^{-\alpha}$$

Directional Sampling

5.1 Random-Ray Monte Carlo

Hit-and-Run Algorithm. For a given current sample $\mathbf{X}_t$, one does the following: (a) uniformly select a random direction $\mathbf{e}_t$, (b) sample a scalar r_t from density $f(r) \propto \pi(\mathbf{X}_t + r\mathbf{e}_t)$, and (c) update $\mathbf{X}_{t+1} = \mathbf{X}_t + r_t\mathbf{e}_t$. This algorithm behaves like a random-direction Gibbs sampler, and it tends to be very helpful if the probability landscape of π consists of distinctive modes along noncoordinate directions.

How to generate the random direction?

- At each step of the algorithm, one has a population of samples
- Randomly select two samples and their difference gives the direction
- If you have the gradient of your distribution use it
- Solving a local optimization problem with a finite number of steps (2 or 3) starting form an initial sample in memory set

Directional Sampling

1. Randomly generates a direction (a unit vector) $\mathbf{e}$.

2. Draws $\mathbf{y}_1, \ldots, \mathbf{y}_k$ from the proposal transition $T_{\mathbf{e}}(\mathbf{x}, \cdot)$ along the direction $\mathbf{e}$. A generic choice is to draw iid samples $r_1, \ldots, r_k$ from $\mathrm{N}(0, \sigma^2)$, where σ can be chosen rather large and set $\mathbf{y}_j = \mathbf{x} + r_j\mathbf{e}$. Another possibility is to draw $r_j \sim \mathrm{Unif}[-\sigma, \sigma]$.

3. Conducts the MTM, as described in Section 3.1.