A Appendix

A.1 Ergodicity of MCMC with auxiliary variables

Without loss of generality assume the current state to be $\boldsymbol{\gamma} \sim \pi(\boldsymbol{\gamma})$. We generate $(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*) \sim q(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*|\boldsymbol{\gamma})$ so that $q(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*|\boldsymbol{\gamma})$ is irreducible and consider $\boldsymbol{\chi}|\boldsymbol{\gamma}, \boldsymbol{\chi}^*, \boldsymbol{\gamma}^* \sim h(\boldsymbol{\chi}|\boldsymbol{\gamma}, \boldsymbol{\chi}^*, \boldsymbol{\gamma}^*)$ for an arbitrary chosen $h(\cdot|\cdot)$. Then $\boldsymbol{\chi}$ and $\boldsymbol{\chi}^*$ are auxiliary variables. If we accept $\boldsymbol{\gamma}^*$ for $\boldsymbol{\gamma}$ with the acceptance probability $r_m(\boldsymbol{\gamma}, \boldsymbol{\chi}; \boldsymbol{\gamma}^*, \boldsymbol{\chi}^*)$, then the Markov chain in the original space Ω_γ is ergodic and has $\pi(\boldsymbol{\gamma})$ as the unique stationary distribution. Furthermore the limiting probability is equal to the stationary distribution and represent the proportion of time spend in the corresponding states during the simulations. We will show the ergodicity by checking that the chain is π-invariant, a-periodic and irreducible in Ω_γ.

► First, we will show that the constructed Markov chain is π-invariant in space Ω_γ. In order to do that we will show that the detailed balance is satisfied by means of adding and integrating out the auxiliary states $\boldsymbol{\chi}$ and $\boldsymbol{\chi}^*$ with respect to the chosen $q(\cdot|\cdot)$ and $h(\cdot|\cdot)$ functions:

$$
\begin{aligned}
&\pi(\boldsymbol{\gamma})A(\boldsymbol{\gamma}; \boldsymbol{\gamma}^*) \\
&= \int_{\Omega_\chi} \int_{\Omega_{\chi^*}} \pi(\boldsymbol{\gamma}) \mathsf{q}(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*|\boldsymbol{\gamma}) h(\boldsymbol{\chi}|\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*, \boldsymbol{\gamma}) r_m(\boldsymbol{\gamma}, \boldsymbol{\chi}; \boldsymbol{\gamma}^*, \boldsymbol{\chi}^*) d\boldsymbol{\chi}^* d\boldsymbol{\chi} \\
&= \int_{\Omega_\chi} \int_{\Omega_{\chi^*}} \pi(\boldsymbol{\gamma}) \mathsf{q}(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*|\boldsymbol{\gamma}) h(\boldsymbol{\chi}|\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*, \boldsymbol{\gamma}) \times \\
&\min\left\{1, \frac{\pi(\boldsymbol{\gamma}^*) \mathsf{q}(\boldsymbol{\gamma}, \boldsymbol{\chi}|\boldsymbol{\gamma}^*) h(\boldsymbol{\chi}^*|\boldsymbol{\gamma}, \boldsymbol{\chi}, \boldsymbol{\gamma}^*)}{\pi(\boldsymbol{\gamma}) \mathsf{q}(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*|\boldsymbol{\gamma}) h(\boldsymbol{\chi}|\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*, \boldsymbol{\gamma})}\right\} d\boldsymbol{\chi}^* d\boldsymbol{\chi} \\
&= \int_{\Omega_\chi} \int_{\Omega_{\chi^*}} \frac{\pi(\boldsymbol{\gamma}^*) \mathsf{q}(\boldsymbol{\gamma}, \boldsymbol{\chi}|\boldsymbol{\gamma}^*) h(\boldsymbol{\chi}^*|\boldsymbol{\gamma}, \boldsymbol{\chi}, \boldsymbol{\gamma}^*)}{\pi(\boldsymbol{\gamma}^*) \mathsf{q}(\boldsymbol{\gamma}, \boldsymbol{\chi}|\boldsymbol{\gamma}^*) h(\boldsymbol{\chi}^*|\boldsymbol{\gamma}, \boldsymbol{\chi}, \boldsymbol{\gamma}^*)} \times \\
&\min\left\{\pi(\boldsymbol{\gamma}) \mathsf{q}(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*|\boldsymbol{\gamma}, \boldsymbol{\chi}) h(\boldsymbol{\chi}|\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*, \boldsymbol{\gamma}), \frac{\pi(\boldsymbol{\gamma}^*) \mathsf{q}(\boldsymbol{\gamma}, \boldsymbol{\chi}|\boldsymbol{\gamma}^*) h(\boldsymbol{\chi}^*|\boldsymbol{\gamma}, \boldsymbol{\chi}, \boldsymbol{\gamma}^*)}{1}\right\} d\boldsymbol{\chi}^* d\boldsymbol{\chi} \\
&= \int_{\Omega_{\chi^*}} \int_{\Omega_\chi} \pi(\boldsymbol{\gamma}^*) \mathsf{q}(\boldsymbol{\gamma}, \boldsymbol{\chi}|\boldsymbol{\gamma}^*) h(\boldsymbol{\chi}^*|\boldsymbol{\gamma}, \boldsymbol{\chi}, \boldsymbol{\gamma}^*) \times \\
&\min\left\{\frac{\pi(\boldsymbol{\gamma}) \mathsf{q}(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*|\boldsymbol{\gamma}, \boldsymbol{\chi}) h(\boldsymbol{\chi}|\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*, \boldsymbol{\gamma})}{\pi(\boldsymbol{\gamma}^*) \mathsf{q}(\boldsymbol{\gamma}, \boldsymbol{\chi}|\boldsymbol{\gamma}^*) h(\boldsymbol{\chi}^*|\boldsymbol{\gamma}, \boldsymbol{\chi}, \boldsymbol{\gamma}^*)}, 1\right\} d\boldsymbol{\chi} d\boldsymbol{\chi}^* \\
&= \int_{\Omega_{\chi^*}} \int_{\Omega_\chi} \pi(\boldsymbol{\gamma}^*) \mathsf{q}(\boldsymbol{\gamma}, \boldsymbol{\chi}|\boldsymbol{\gamma}^*) h(\boldsymbol{\chi}^*|\boldsymbol{\gamma}, \boldsymbol{\chi}, \boldsymbol{\gamma}^*) r_m(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*; \boldsymbol{\gamma}, \boldsymbol{\chi}) d\boldsymbol{\chi} d\boldsymbol{\chi}^* \\
&= \pi(\boldsymbol{\gamma}^*) A(\boldsymbol{\gamma}^*; \boldsymbol{\gamma}).
\end{aligned}
$$

Note that in the derivation above change of the order of integration is always possible according to Fubini's theorem, since all of the addressed probability measures are measurable and bounded by definition. The construction of the addressed Markov chain is such that based on the acceptance probability there always is a positive probability of not changing the state, providing the required a-periodicity in Ω_γ, whilst the irreducibility in Ω_γ is guaranteed by the way we construct $q(\boldsymbol{\gamma}^*, \boldsymbol{\chi}^*|\boldsymbol{\gamma})$. Thus, we have shown the Markov chain to be a-periodic, irreducible and π-invariant in Ω_γ, which is sufficient for its ergodicity in Ω_γ. ◄

A.2 Mode jumping MCMC algorithm

Below the detailed MJMCMC algorithm, a step of which is described in the article, is represented in form of pseudo-code. Let

- ϱ - the probability deciding if a large jump with local optimization is used;
- $P_f(\cdot)$ - the distribution for the choice of the local optimizers;
- $P_l(\cdot)$ - the distribution for the choice of large jump transition kernel;
- $P_r(\cdot)$ - the distribution for the choice of the randomizing kernel;
- $f_i(\cdot), i \in \{1, ..., I\}$ - the set of local optimizers;
- $\mathsf{q}_l^{(j)}(\cdot|\cdot), j \in \{1, ..., J\}$ - the set of large jump kernels;
- $\mathsf{Q}_\mathsf{o}^{(j)}(\cdot|\cdot), j \in \{1, ..., J\}$ - the set of optimization proposal kernels;
- $\mathsf{q}_r^{(j)}(\cdot|\cdot), j \in \{1, ..., J\}$ - the set of randomizing kernels.

Then pseudo-code of MJMCMC looks as follows:

Algorithm 1 Mode jumping MCMC

procedure MJMCMC(N, $\varrho, P_m(\cdot), P_f(\cdot), P_l(\cdot), P_o(\cdot), P_l(\cdot), P_p(\cdot)$)
  $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma}_0$ ▷ define the initial state
  for $t \in \{1, ..., N\}$ **do**
    $v \sim Unif[0; 1]$
    if $v \leq \varrho$ **then** ▷ perform a large jump with local optimization
      $\mathsf{q}_l \sim P_l(\boldsymbol{\zeta})$ ▷ choose large jump kernel
      $f \sim P_f(\boldsymbol{\zeta})$ ▷ choose local optimizer
      $\mathsf{q}_r \sim P_r(\boldsymbol{\zeta})$ ▷ choose randomization kernel
      $\boldsymbol{\chi}_0^* \sim \mathsf{q}_l(\boldsymbol{\zeta}|\boldsymbol{\gamma})$ ▷ make a large jump
      $\boldsymbol{\chi}_\mathsf{o}^* \leftarrow f(\boldsymbol{\chi}_0^*)$ ▷ perform local optimization
      $\boldsymbol{\gamma}^* \sim \mathsf{q}_r(\boldsymbol{\zeta}|\boldsymbol{\chi}_\mathsf{o}^*)$ ▷ make randomization around the mode
      $\boldsymbol{\chi}_0 \sim \mathsf{q}_l(\boldsymbol{\zeta}|\boldsymbol{\gamma}^*)$ ▷ make a reverse symmetric large jump
      $\boldsymbol{\chi}_\mathsf{o} \leftarrow f(\boldsymbol{\chi}_0)$ ▷ perform local optimization
      $u \sim Unif[0; 1]$
      if $u \leq r_m(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*)$ **then** ▷ $r_m(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*)$ is eq. (21) in the article.
        $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma}^*$ ▷ accept the move
      end if
    else
      $\boldsymbol{\gamma} \leftarrow f_0(\boldsymbol{\gamma})$ ▷ otherwise make an MTMCMC step
    end if
  end for
end procedure

A.3 Multiple try MCMC algorithm

Multiple-try Metropolis is a sampling method that is a modified form of the Metropolis-Hastings method, designed to be able to properly parallelize the original Metropolis-Hastings algorithm. The idea of the method is to allow generating S trial proposals $\boldsymbol{\chi}_1^*, ...\boldsymbol{\chi}_S^*$ in parallel from the proposal $q(\cdot|\cdot)$, which is initially selected from a set of proposals with respect to the distribution for the choice of the MTMCMC proposal $P_p(\cdot)$. Then within a trial set $\boldsymbol{\chi}^* \in \{\boldsymbol{\chi}_1^*, ..., \boldsymbol{\chi}_S^*\}$ is selected with probability proportional to some importance weights $w(\boldsymbol{\chi}, \boldsymbol{\chi}_i^*), i \in \{1, ..., S\}$. In the reversed move $\boldsymbol{\chi}_1, ...\boldsymbol{\chi}_{S-1}$ are generated conditioning on $\boldsymbol{\chi}^*$ from the proposal $q(\boldsymbol{\chi}|\boldsymbol{\chi}^*)$ and $\boldsymbol{\chi}_S = \boldsymbol{\chi}$. Finally, the move is accepted with probability $r_m(\boldsymbol{\chi}, \boldsymbol{\chi}^*)$. A step of MTMCMC algorithm ($N = 1$) is addressed as $f_0(\cdot)$ in the pseudo-code of MJMCMC and we recommend that it is addressed in at least 95% of the iterations of MJMCMC. Simultaneously $N > 1$ steps of MTMCMC can be seen as a local combinatorial optimization procedure and thus are used as one of the optimizers in MJMCMC then this procedure is addressed in the pseudo-code above as $f_4(\cdot)$.

Algorithm 2 Multiple try MCMC

procedure MTMCMC(N, $\lambda(\cdot,\cdot), P_p(\cdot), \boldsymbol{\chi}_0, S$) ▷ N - number of steps after burn-in, $\omega(\cdot,\cdot)$ - the weight function, $P_p(\cdot)$ - distribution for choice of the proposal kernel, $\boldsymbol{\chi}_0$ - initial state, and S - size of the proposed sample.
  $\mathfrak{q} \sim P_p(\boldsymbol{\zeta})$ ▷ choose the MTMCMC proposal
  for $t \in \{1, ..., N\}$ **do**
    $\boldsymbol{\chi}_1^*, ..., \boldsymbol{\chi}_S^* \sim q(\boldsymbol{\gamma}|\boldsymbol{\chi})$ ▷ pick S neighbors of the current state
    $\boldsymbol{\chi}^* \sim \omega(\boldsymbol{\chi})$ ▷ pick one of the proposals with respect to weights
    $\boldsymbol{\chi}_1, ..., \boldsymbol{\chi}_{S-1} \sim q(\boldsymbol{\gamma}|\boldsymbol{\chi}^*)$ ▷ pick $S-1$ random neighbors of $\boldsymbol{\chi}$
    $\boldsymbol{\chi}_S \leftarrow \boldsymbol{\chi}$ ▷ define the last one as the current state
    $u \sim Unif[0; 1]$
    if $u \leq r_m(\boldsymbol{\chi}, \boldsymbol{\chi}^*)$ **then** ▷ $r_m(\boldsymbol{\chi}, \boldsymbol{\chi}^*)$ is of form (23)
      $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}^*$ ▷ accept the move
    end if
  end for
  return $\boldsymbol{\gamma}$
end procedure

A.4 Simulated annealing algorithm

Simulated annealing algorithm is used to suggest the locally annealed proposals for MJMCMC. It is based on the idea that the acceptance probabilities $p_a(x, y|t) = \min\left\{1, e^{\frac{\mathsf{G}(y)-\mathsf{G}(x)}{t}}\right\}$ of the moves depend not only on the objective function values $G(x)$ and $G(y)$ but also on the temperature parameter t. This allows to accept deteriorating solutions at the beginning of the procedure (when $t > 1$), but once $t < 1$ the algorithm becomes greedy and the search is intensified. The temperatures are changed with respect to the annealing schedule T_c, which is often considered to be exponential, namely $t_{i+1} = t_i e^{-\Delta t}$. For a given tem-

Algorithm 3 Simulated annealing optimization

procedure SIMULATEDANNEAL($T_c, p_a(.,.|.), G(\cdot), P_p(\cdot), \mathbb{N}(\cdot), \boldsymbol{\chi}_0$) ▷ T_c - cooling schedule, $p_a(.,.|.)$ - acceptance probabilities, $G(\cdot)$ - objective function, $\mathbb{N}(\cdot)-$neighborhood defined by transition kernel $q(\cdot|\cdot)$ chosen from the distribution for the choice of the SA proposal $P_p(\cdot)$, $\boldsymbol{\chi}_0$ - initial state.
  $\mathsf{q} \sim P_p(\boldsymbol{\zeta})$ ▷ choose the SA proposal
  $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_0$
  $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_0$
  for $t \in T_c$ **do** ▷ for all temperatures in the cooling schedule
    $\boldsymbol{\chi}_c \leftarrow \mathbb{N}(\boldsymbol{\chi})$ ▷ pick a random neighbor of the current solution
    if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_b)$ **then**
      $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_c$ ▷ update the best found solution
    end if
    $u \sim Unif[0; 1]$
    if $u \leq p_a(\boldsymbol{\chi}, \boldsymbol{\chi}_c|t)$ **then**
      $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$ ▷ accept the move
    end if
  end for
  return $\boldsymbol{\chi}, \boldsymbol{\chi}_b$ ▷ return the final solution
end procedure

perature t the algorithm is generating an ergodic Markov chain and converges to the stationary limiting distribution $\pi_t(x) = \frac{1}{N_0(t)} e^{\frac{\mathsf{G}(x)}{t}}$, where the normalizing constants $N_0(t)$ are computed as $N_0(t) = \sum_{x\in\Omega_x} e^{\frac{\mathsf{G}(x)}{t}}$. Standard Metropolis-Hastings algorithm is a particular case of simulated annealing algorithm when $t = 1$, $q(y|x)$ is symmetric, and $G(x) = \log \pi(x)$, where $\pi(x)$ is the target distribution of the corresponding parameter of interest. The acceptance then ratio becomes $p_a(x, y|t = 1) = \min\left\{1, e^{\log \pi(y) - \log \pi(x)}\right\} = \min\left\{1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\right\}$, which exactly corresponds to the Metropolis-Hastings acceptance probabilities, for $t = 1$ correspondingly $\pi_t(x) = \frac{e^{\frac{\mathsf{G}(x)}{t}}}{\sum_{x\in\Omega_x} e^{\frac{\mathsf{G}(x)}{t}}} = \pi(x)$. This allows to use the sequences obtained from the simulated annealing at $t = 1$ as samples from the target distribution right away. Limiting probabilities for the other temperatures can be linked to the target distributions of Metropolis-Hastings as $\pi(x) \propto [\pi_t(x)]^{\frac{1}{t}}$, which allows to combine the auxiliary states of simulated annealing procedure with the MJMCMC for the inference on the target distribution. Simulated annealing procedure is marked as $f_1(\cdot)$ in MJMCMC.

A.5 Accept the first improving neighbor algorithm

In this subsection we describe construction of another algorithm used to suggest the locally annealed MCMC proposals, namely accept the first improving neighbor optimization. The algorithm is based on the idea that for the current solution x we make a transition to the first improving neighbor $y \in \mathbb{N}(x)$, such that $G(y) > G(x)$. If the local stop parameter (ls) is true, then condition $G(y) \leq G(x), \forall y \in \mathbb{N}(x)$ would be the stopping criterion, otherwise the maximum in the neighborhood $\mathbb{N}(x)$ is selected as the new current solution, namely $x \leftarrow \text{argmax}_{y \in \mathbb{N}(x)} G(y)$ and the procedure is continued for the predefined number of iterations S. Accept the first improving neighbor algorithm is marked as $f_2(\cdot)$ in the pseudo-code of MJMCMC above.

Algorithm 4 Accept the first improving optimization

procedure GREEDYFIRST($G(\cdot), \mathbb{N}(\cdot), q(\cdot|\cdot), ls, S, \boldsymbol{\chi}_0$) ▷ $G(\cdot)$ - objective function, $\mathbb{N}(\cdot)$ - neighborhood defined by $q(\cdot|\cdot)$ chosen from the distribution for the choice of the proposal $P_p(\cdot)$, S - number of steps, ls - local stop criterion, $\boldsymbol{\chi}_0$ - initial point.
    $\mathsf{q} \sim P_p(\boldsymbol{\zeta})$ ▷ choose the proposal
    $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_0$
    $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_0$
    $i \leftarrow 0$
    for $i \in \{1, ..., S\}$ **do** ▷ for all iterations or until the stop criteria is met
        $\boldsymbol{\chi}_n \leftarrow null$
        $\mathsf{G}(\boldsymbol{\chi}_n) \leftarrow -\infty$
        for all $\boldsymbol{\chi}_c \in \mathbb{N}(\boldsymbol{\chi})$ **do** ▷ for all neighbors
            if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_b)$ **then**
                $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_c$ ▷ update the best found solution
                $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$
                $\boldsymbol{\chi}_n \leftarrow \boldsymbol{\chi}_c$ **break for**
            else
                if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_n)$ **then**
                    $\boldsymbol{\chi}_n \leftarrow \boldsymbol{\chi}_c$ ▷ update the best neighboring solution
                    $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$
                end if
            end if
        end for
        if ls and $\boldsymbol{\chi}_n <> \boldsymbol{\chi}_b$ **then**
            $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_b$ **break for**
        else
        end if
    end for
    return $\boldsymbol{\chi}, \boldsymbol{\chi}_b$ ▷ return the final solution
end procedure

A.6 Accept the best neighbor algorithm

In this subsection we describe construction of the greedy heuristic algorithm known as accept the best neighbor optimization. The algorithm is similar to the previous one and is based on the idea that for the current solution x we make a transition to the best $y \in \mathbb{N}(x)$, such that $G(y) > G(x)$. If the ls is true and $G(y) \leq G(x), \forall y \in \mathbb{N}(x)$ the algorithm is stopped, otherwise the maximum in the neighborhood $\mathbb{N}(x)$ is selected as the new current solution, namely $x \leftarrow \text{argmax}_{y \in \mathbb{N}(x)} G(y)$ and the procedure is continued for the predefined number of iterations S. Accept the best neighbor algorithm is marked as procedure $f_3(\cdot)$ in the pseudo-code of MJMCMC above.

Algorithm 5 Accept the best optimization

procedure GREEDYBEST($G(\cdot), \mathbb{N}(\cdot), q(\cdot|\cdot), ls, S, \boldsymbol{\chi}_0$) ▷ $G(\cdot)$ - objective function, $\mathbb{N}(\cdot)$ - neighborhood defined by $q(\cdot|\cdot)$ chosen from the distribution for the choice of the proposal $P_p(\cdot)$, S - number of steps, ls - local stop criterion, $\boldsymbol{\chi}_0$ - initial point.
  $\mathsf{q} \sim P_p(\boldsymbol{\zeta})$ ▷ choose the proposal
  $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_0$
  $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_0$
  $i \leftarrow 0$
  for $i \in \{1, ..., S\}$ **do** ▷ for all iterations or until the stop criteria is met
    $\boldsymbol{\chi}_n \leftarrow null$
    $\mathsf{G}(\boldsymbol{\chi}_n) \leftarrow -\infty$
    for all $\boldsymbol{\chi}_c \in \mathbb{N}(\boldsymbol{\chi})$ **do** ▷ for all neighbors
      if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_b)$ **then**
        $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_c$ ▷ update the best found solution
        $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$
        $\boldsymbol{\chi}_n \leftarrow \boldsymbol{\chi}_c$
      else
        if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_n)$ **then**
          $\boldsymbol{\chi}_n \leftarrow \boldsymbol{\chi}_c$ ▷ update the best neighboring solution
          $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$
        end if
      end if
    end for
    if ls and $\boldsymbol{\chi}_n <> \boldsymbol{\chi}_b$ **then**
      $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_b$ **break for**
    end if
  end for
  return $\boldsymbol{\chi}, \boldsymbol{\chi}_b$ ▷ return the final solution
end procedure

References

J. S. Liu, F. Liang, and W. H. Wong. The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *Acm computing surveys*, pages 268–308, 2003.

H. Tjelmeland and B. K. Hegstad. Mode jumping proposals in MCMC. *Scandinavian journal of statistics*, 28:205–223, 1999.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

G. Storvik. On the flexibility of Metropolis-Hastings acceptance probabilities in auxiliary variable proposal generation. *Scandinavian Journal of Statistics*, 38:342–358, 2011.