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.

Algorithm 1 Mode jumping MCMC

```
1: procedure MJMCMC($N$, $P_c(\cdot,\cdot)$, $f_i(\cdot)$, $\mathsf{q}_l^{(i,j)}(\cdot|\cdot)$, $\mathsf{Q}_\mathsf{o}^{(i,j)}(\cdot|\cdot)$, $\mathsf{q}_r^{(i,j)}(\cdot|\cdot)$, $\varrho^{(i,j)}$)
   ▷ $N$ - number of iterations of MJMCMC, $P_c(\cdot,\cdot)$ - kernel for the choice of the local optimizers and transition kernel type, $f_i(\cdot)$ - the set of local optimizers, $\mathsf{q}_l^{(i,j)}(\cdot|\cdot)$ - large jump kernels, $\mathsf{Q}_\mathsf{o}^{(i,j)}(\cdot|\cdot)$ - optimization proposal kernels, $\mathsf{q}_r^{(i,j)}(\cdot|\cdot)$ - randomizing kernels, $\varrho^{(i,j)}$ - other parameters of local optimizers. Notice that $i \in \{0,...,I\}$ and $j \in \{1,...,J\}$.
2:     $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma}_0$    ▷ define the initial point in space $\Omega_\gamma$
3:     burn-in    ▷ carry out burn-in, learn about $P_c(i,j)$, namely the proportions of time local optimization $i$ is used with the kernel type $j$
4:     for $t \in \{1,...,N\}$ do    ▷ proceed with MJMCMC for $N$ iterations
5:         $\{n,k\} \leftarrow P_c(x,y)$    ▷ chose the type of local optimizer $n > 0$ (or no local optimizer $n = 0$) and the corresponding kernel type $k$ for it
6:         if $n > 0$ then    ▷ carry out local optimization
7:             $\boldsymbol{\chi}_0^* \leftarrow \mathsf{q}_l^{(n,k)}(\boldsymbol{\zeta}|\boldsymbol{\gamma})$    ▷ make a large jump
8:             $\boldsymbol{\chi}_\mathsf{o}^* \leftarrow f_n(\boldsymbol{\chi}_0^*, \mathsf{Q}_\mathsf{o}^{(n,k)}(\cdot|\cdot), \varrho^{(n,k)})$    ▷ perform local optimization
9:             $\boldsymbol{\gamma}^* \leftarrow \mathsf{q}_r^{(n,k)}(\boldsymbol{\zeta}|\boldsymbol{\chi}_\mathsf{o}^*)$    ▷ make randomization around the mode
10:            $\boldsymbol{\chi}_0 \leftarrow \mathsf{q}_l^{(n,k)}(\boldsymbol{\zeta}|\boldsymbol{\gamma}^*)$    ▷ make a reverse symmetric large jump
11:            $\boldsymbol{\chi}_\mathsf{o} \leftarrow f_n(\boldsymbol{\chi}_0, \mathsf{Q}_\mathsf{o}^{(n,k)}(\cdot|\cdot), \varrho^{(n,k)})$    ▷ perform local optimization
12:            $\mathsf{q}_r^{(n,k)}(\boldsymbol{\gamma}|\boldsymbol{\chi}_\mathsf{o})$    ▷ find the probability of a transition from the obtained mode to the current solution
13:            if $r_m(\boldsymbol{\gamma},\boldsymbol{\gamma}^*) = \min\left\{1, \frac{\pi(\boldsymbol{\gamma}^*)\mathsf{q}_\mathsf{r}^{(n,k)}(\boldsymbol{\gamma}|\boldsymbol{\chi}_k)}{\pi(\boldsymbol{\gamma})\mathsf{q}_\mathsf{r}^{(n,k)}(\boldsymbol{\gamma}^*|\boldsymbol{\chi}_k^*)}\right\} \geq u \sim Unif[0;1]$ then
14:                $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma}^*$    ▷ accept the move $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma}^*$ with respect to the current step's acceptance probability or remain in the old state $\boldsymbol{\gamma}$ otherwise
15:            end if
16:        else $\boldsymbol{\gamma} \leftarrow f_0((\boldsymbol{\gamma}, \mathsf{Q}_\mathsf{o}^{(0,k)}(\cdot|\cdot), \varrho^{(0,k)})$ ▷ if no local optimization is chosen ($n = 0$) - make an MTMCMC step
17:        end if
18:        $\mathbb{V} \leftarrow \mathbb{V} \uplus \{\boldsymbol{\gamma}^*, \boldsymbol{\chi}, \boldsymbol{\chi}^*\}$    ▷ append uniquely the newly visited solutions to the set of visited solutions
19:    end for
20: 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. 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 ($N = 1$, no burn-in) 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 $N = n, n > 1$ 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 (n steps of MTMCMC) is addressed in the pseudo-code above as $f_4(\cdot)$.

Algorithm 2 Multiple try MCMC

```
1: procedure MTMCMC($N$, $\lambda(\cdot,\cdot), q(\cdot|\cdot), \boldsymbol{\chi}_0, S$)    ▷ $N$ - number of steps after burn-in, $\lambda(\cdot,\cdot)$ - symmetric lambda function, $q(\cdot|\cdot)$ - proposal kernel, $\boldsymbol{\chi}_0$ - initial state, and $S$ - size of the proposed sample (number of cores).
2:     $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_0$
3:     if $N > 1$ then
4:         burn-in    ▷ carry out burn-in if necessary
5:     end if
6:     for $t \in \{1, ..., N\}$ do    ▷ perform MTMCMC for the predefined number of iterations
7:         $\boldsymbol{\chi}_1^*, ...., \boldsymbol{\chi}_S^* \sim q(\boldsymbol{\gamma}|\boldsymbol{\chi})$    ▷ pick $S$ random neighbors of the current solution
8:         $\boldsymbol{\chi}^* \sim K \times \omega(\boldsymbol{\chi}, \boldsymbol{\chi}^*) = K \times \pi(\boldsymbol{\chi}_i^*)q(\boldsymbol{\chi}|\boldsymbol{\chi}_i^*)\lambda(\boldsymbol{\chi}, \boldsymbol{\chi}_i^*), i \in \{1, ..., S\}$    ▷ pick one of the proposals with respect to probabilities proportional to their weights
9:         $\boldsymbol{\chi}_1, ...., \boldsymbol{\chi}_{S-1} \sim q(\boldsymbol{\gamma}|\boldsymbol{\chi}^*)$    ▷ pick $S - 1$ random neighbors of the selected proposal and let $\boldsymbol{\chi}_S = \boldsymbol{\chi}$
10:        if $r_m(\boldsymbol{\chi}, \boldsymbol{\chi}^*) = \min\left\{1, \frac{w(\boldsymbol{\chi}, \boldsymbol{\chi}_1^*) + \cdots + w(\boldsymbol{\chi}, \boldsymbol{\chi}_S^*)}{w(\boldsymbol{\chi}^*, \boldsymbol{\chi}_1) + \cdots + w(\boldsymbol{\chi}^*, \boldsymbol{\chi}_S)}\right\} \geq u \sim Unif[0; 1]$ then
11:            $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}^*$    ▷ accept the move with respect to the acceptance probability or remain in the old state otherwise
12:        end if
13:    end for
14:    return $\boldsymbol{\gamma}$
15: 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 (whilst $t > 1$) and diversify the search, whilst once $t < 1$ the algorithm becomes greedy and intensifies the search. The temperatures are changed with respect to the annealing schedule T_c, which is often considered to be exponential, namely $t_{i+1} = t_i e^{-\Delta t}$. Notice that for a given temperature t the algorithm is generating

Algorithm 3 Simulated annealing optimization

```
1: procedure SimulatedAnneal($T_c, p_a(.,.|.), G(\cdot), q(\cdot|\cdot), \mathbb{N}(\cdot), \boldsymbol{\chi}_0$)    ▷ $T_c$ - cooling schedule, $p_a(.,.|.)$ - acceptance probabilities, $G(\cdot)$ - objective function, $\mathbb{N}(\cdot)-$neighborhood defined by transition kernel $q(\cdot|\cdot)$, $\boldsymbol{\chi}_0$ - initial state.
2:     $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_0$
3:     $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_0$
4:     for $t \in T_c$ do    ▷ for all temperatures in the cooling schedule
5:         $\boldsymbol{\chi}_c \leftarrow \mathbb{N}(\boldsymbol{\chi})$    ▷ pick a random neighbor of the current solution
6:         if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_b)$ then
7:             $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_c$    ▷ update the best found solution
8:         end if
9:         if $p_a(\boldsymbol{\chi}, \boldsymbol{\chi}_c|t) > u \sim Unif[0;1]$ then
10:            $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$    ▷ accept the move with some probability
11:        end if
12:    end for
13:    return $\boldsymbol{\chi}, \boldsymbol{\chi}_b$    ▷ return the final solution
14: end procedure
```

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 $N_0(t) = \sum_{x\in\Omega_x} e^{\frac{\mathsf{G}(x)}{t}}$. It is interesting that the standard Metropolis-Hastings algorithm is a particular case of simulated annealing algorithm when $t = 1$ and $G(x) = \log \pi(x) q(y|x)$, where $\pi(x)$ is the target distribution of the corresponding parameter of interest and $q(y|x)$ is the proposal, then the acceptance ratio becomes $p_a(x,y|t=1) = \min\left\{1, e^{\log \pi(y)q(x|y) - \log \pi(x)q(y|x)}\right\} = \min\left\{1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\right\}$, 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 in principle allows to use the sequences obtained from the simulated annealing at $t = 1$ as samples from the target distribution right away, whilst the 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. This procedure is marked as $f_1(\cdot)$ in the pseudo-code of 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

```
 1: 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 $q(\cdot|\cdot)$, $S$ - number of steps, $ls$ - local stop criterion, $\boldsymbol{\chi}_0$ - initial point.
 2:     $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_0$
 3:     $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_0$
 4:     $i \leftarrow 0$
 5:     for $i \in \{1, ..., S\}$ do    ▷ for all iterations or until the stop criteria is met
 6:         $\boldsymbol{\chi}_n \leftarrow null$
 7:         $\mathsf{G}(\boldsymbol{\chi}_n) \leftarrow -\infty$
 8:         for all $\boldsymbol{\chi}_c \in \mathbb{N}(\boldsymbol{\chi})$ do    ▷ iteration through the neighbors of the given solution
 9:             if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_b)$ then
10:                 $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_c$    ▷ update the best found solution
11:                 $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$
12:                 $\boldsymbol{\chi}_n \leftarrow \boldsymbol{\chi}_c$ break for
13:             else
14:                 if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_n)$ then
15:                     $\boldsymbol{\chi}_n \leftarrow \boldsymbol{\chi}_c$    ▷ update the best neighboring solution
16:                     $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$
17:                 end if
18:             end if
19:         end for
20:         if $ls$ and $\boldsymbol{\chi}_n <> \boldsymbol{\chi}_b$ then
21:             $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_b$ break for
22:         else
23:         end if
24:     end for
25:     return $\boldsymbol{\chi}, \boldsymbol{\chi}_b$    ▷ return the final solution
26: 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

```
 1: 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 $q(\cdot|\cdot)$, $S$ - number of steps, $ls$ - local stop criterion, $\boldsymbol{\chi}_0$ - initial point.
 2:     $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_0$
 3:     $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_0$
 4:     $i \leftarrow 0$
 5:     for $i \in \{1, ..., S\}$ do   ▷ for all iterations or until the stop criteria is met
 6:         $\boldsymbol{\chi}_n \leftarrow null$
 7:         $\mathsf{G}(\boldsymbol{\chi}_n) \leftarrow -\infty$
 8:         for all $\boldsymbol{\chi}_c \in \mathbb{N}(\boldsymbol{\chi})$ do   ▷ iteration through the neighbors of the given solution
 9:             if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_b)$ then
10:                 $\boldsymbol{\chi}_b \leftarrow \boldsymbol{\chi}_c$   ▷ update the best found solution
11:                 $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$
12:                 $\boldsymbol{\chi}_n \leftarrow \boldsymbol{\chi}_c$
13:             else
14:                 if $\mathsf{G}(\boldsymbol{\chi}_c) > \mathsf{G}(\boldsymbol{\chi}_n)$ then
15:                     $\boldsymbol{\chi}_n \leftarrow \boldsymbol{\chi}_c$   ▷ update the best neighboring solution
16:                     $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_c$
17:                 end if
18:             end if
19:         end for
20:         if $ls$ and $\boldsymbol{\chi}_n <> \boldsymbol{\chi}_b$ then
21:             $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}_b$ break for
22:         end if
23:     end for
24:     return $\boldsymbol{\chi}, \boldsymbol{\chi}_b$   ▷ return the final solution
25: end procedure
```