Rare event simulation for energy systems and power network

Debarati Bhaumik

Jointly worked with

Daan Crommelin & Bert Zwart

Centrum Wiskunde & Informatica

Supported by

Shell

FOM

NWO Netherlands Organisation for Scientific Research

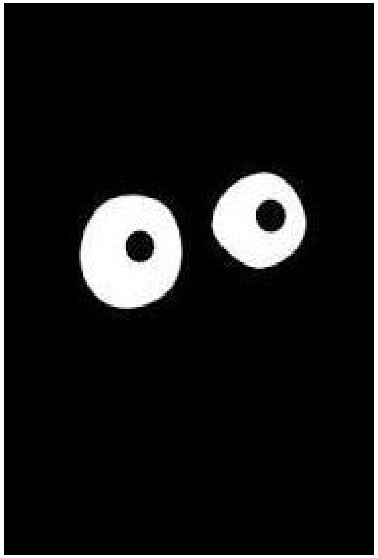
Mitigation of large power spills in stand-alone energy system with wind generation and storage

What is power spill?

Power generation > Power demand

Why to mitigate large power spills?

Excess power → Grid constraint violations → Physical damage to grid



What causes power spills?

What causes power spills?

Integration of renewable energy sources.

Photo-voltaic arrays

Wind turbines

What causes power spills?

Integration of renewable energy sources.

Photo-voltaic arrays

Wind turbines

Unpredictable nature → Power imbalances → Power spill

How can energy storage help?

How can energy storage help?

1. Energy storage devices act as buffer.

2. Act as peak-shavers.

What do we want?

To find the best way to operate the battery such that *probability of large power spill* is minimal for the energy system.

Probability of large power spill (PLPS)

- Power generation > Power demand.
- Battery cannot absorb all the excess power generated due to *battery constraints*.
- $F(t)$: Residual power.
- $F(t) > 0$ is **power spill**.

Probability of large power spill (PLPS)

- Power generation > Power demand.
- Battery cannot absorb all the excess power generated due to *battery constraints*.
- F(t) : Residual power.
- F(t) > 0 is **power spill**.

Large power spill : $F(t) \geq F_0$ (>0 large power spill threshold)

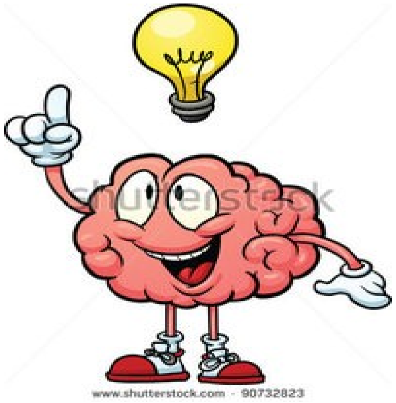
PLPS calculates the probability of large power spill in the system over period *T*

$$\gamma = P\left(\{ \sup_{t \in [0,T]} F(t)\} \geq F_0\right)$$

Main challenge

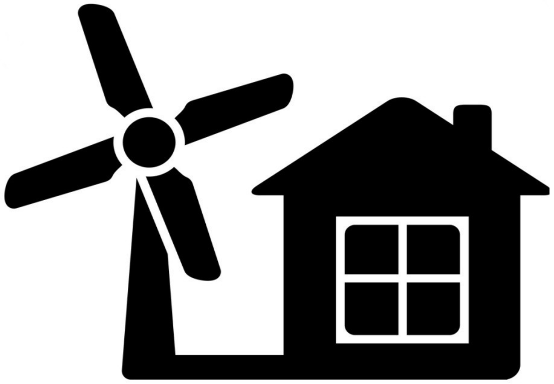
Calculating small values of PLPS using **Crude Monte Carlo** simulation is expensive.

Solution

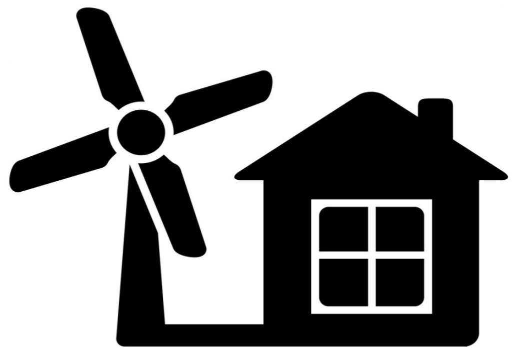


Use **splitting** technique for rare-event simulations to reduce the workload.

System setup



System setup...



Power injections

- Stochastic wind power generation: W(t)
- Stochastic power demand: D(t)
- Power mismatch: $P(t) = W(t) – D(t)$.

System setup...

Battery model

The battery is charged according to:

$$B\left(t+\Delta t\right)=B\left(t\right)+p^{B}\left(t\right)\Delta t \qquad (1)$$

$p^{B}(t)$: power flowing in/out of the battery and is related to $P(t)$ by the battery constraints :

1. *Capacity constraint* : $\quad 0\leq B\left(t\right)\leq B_{max} \qquad (2)$

2. *Ramp constraint* : $\quad -\beta\leq\frac{B\left(t+\Delta t\right)-B\left(t\right)}{\Delta t}\leq\beta \qquad (3)$

Why use Splitting and not CMC?

A = ***rare event set*** of interest.
$\gamma = P(A)$

Why use Splitting and not CMC?

A = ***rare event set*** of interest.
$\gamma = P(A)$

Crude Monte Carlo (CMC)

Why use Splitting and not CMC?

A = ***rare event set*** of interest.
$\gamma = P(A)$

Crude Monte Carlo (CMC)

- Computes: $\gamma = \dfrac{1}{n}\sum_{j=1}^{n} 1_{[A \in j]}$

Why use Splitting and not CMC?

A = ***rare event set*** of interest.
$\gamma = P(A)$

Crude Monte Carlo (CMC)

- Computes: $\gamma = \dfrac{1}{n}\sum_{j=1}^{n} 1_{[A \in j]}$

- *Squared relative error* : SRE(γ) = $(1 - \gamma)/(\gamma\, n)$.

Why use Splitting and not CMC?

A = ***rare event set*** of interest.
$\gamma = P(A)$

Crude Monte Carlo (CMC)

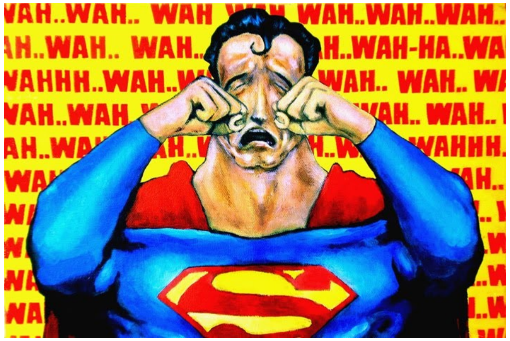
- Computes: $\gamma = \dfrac{1}{n}\sum_{j=1}^{n} 1_{[A \in j]}$
- *Squared relative error* : SRE(γ) = (1 – γ)/(γ *n*).
- SRE(γ) $\rightarrow \infty$ as γ $\rightarrow 0$

Why use Splitting and not CMC?

A = ***rare event set*** of interest.
$\gamma = P(A)$

Crude Monte Carlo (CMC)

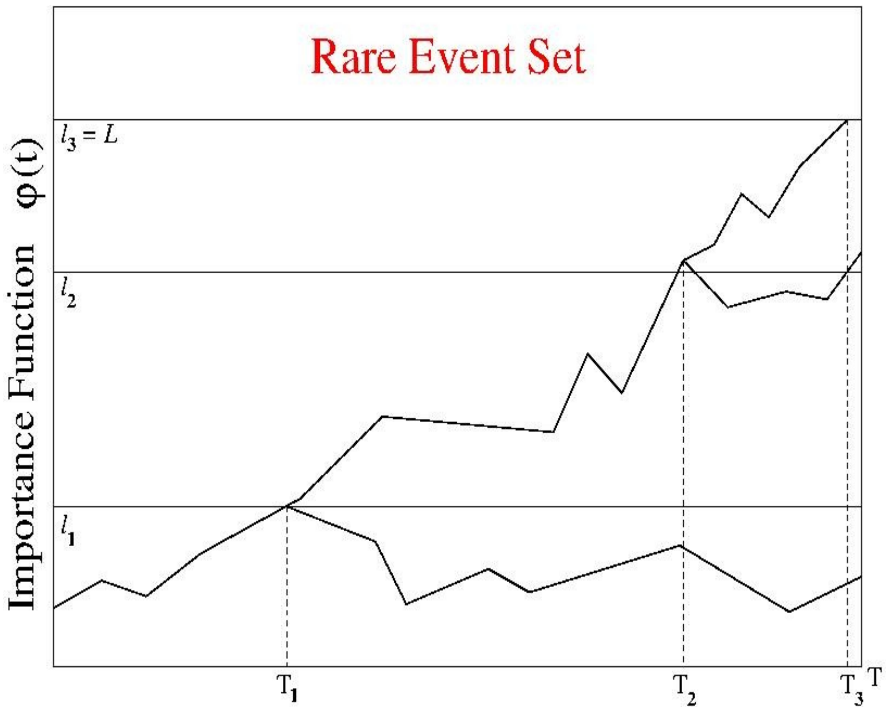
- Computes: $\gamma = \dfrac{1}{n}\sum_{j=1}^{n} 1_{[A \in j]}$
- *Squared relative error* : SRE(γ) = $(1 - \gamma)/(\gamma\, n)$.
- SRE(γ) $\to \infty$ as $\gamma \to 0$
- CMC gets computationally very expensive!



Splitting technique

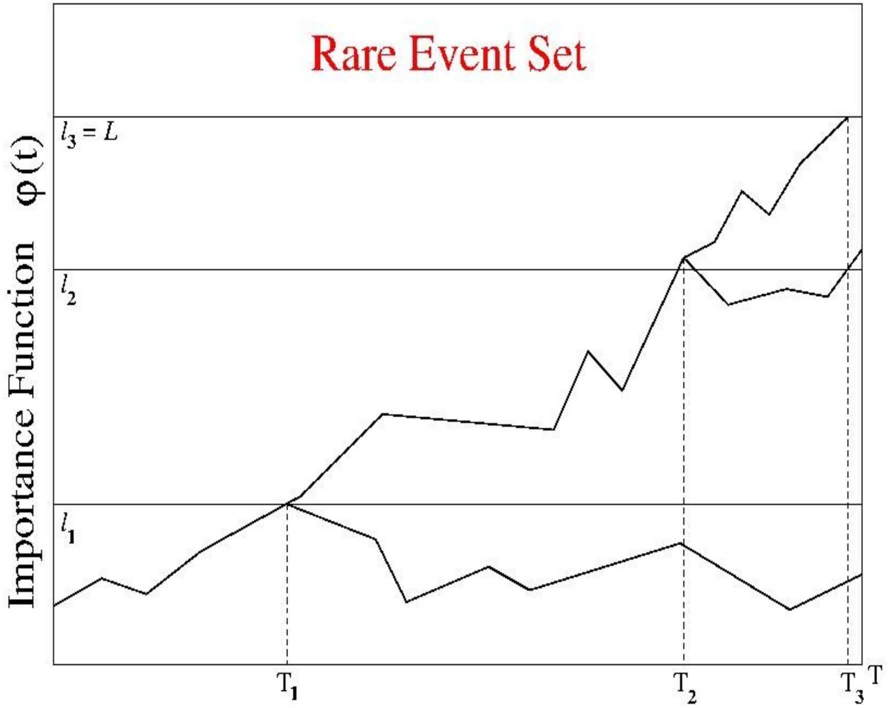
Splitting technique

- *Importance Function* (IF) measures distance to A.



Splitting technique

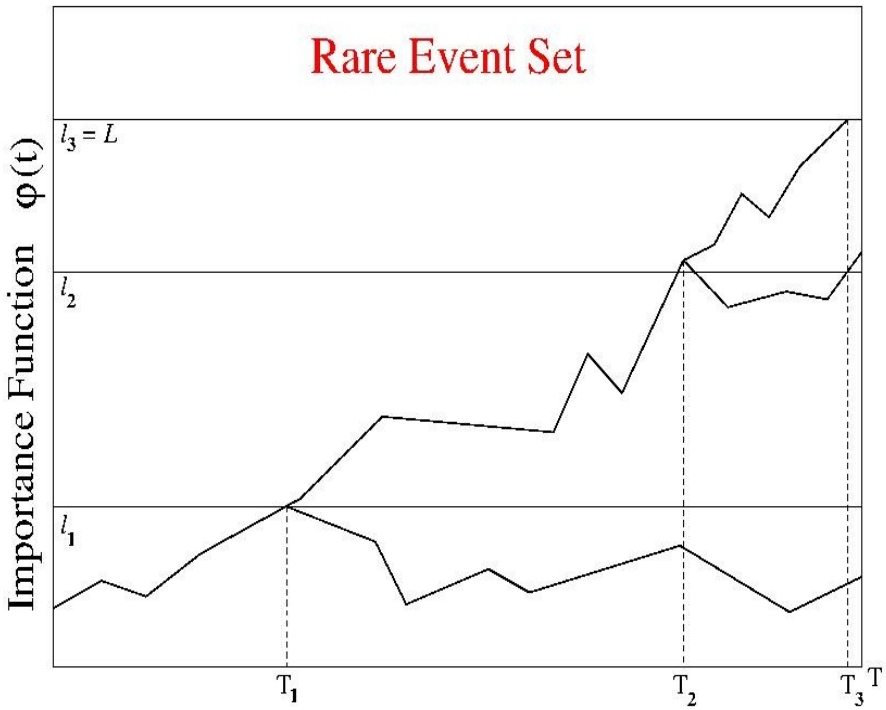
- *Importance Function* (IF) measures distance to A.



- Decompose distance to A into various '**non-rare**' levels of the IF.

Splitting technique

- *Importance Function* (IF) measures distance to A.
- Decompose distance to A into various '**non-rare**' levels of the IF.
- Sample paths of stochastic processes involved split into multiple copies at various IF levels till A is reached.



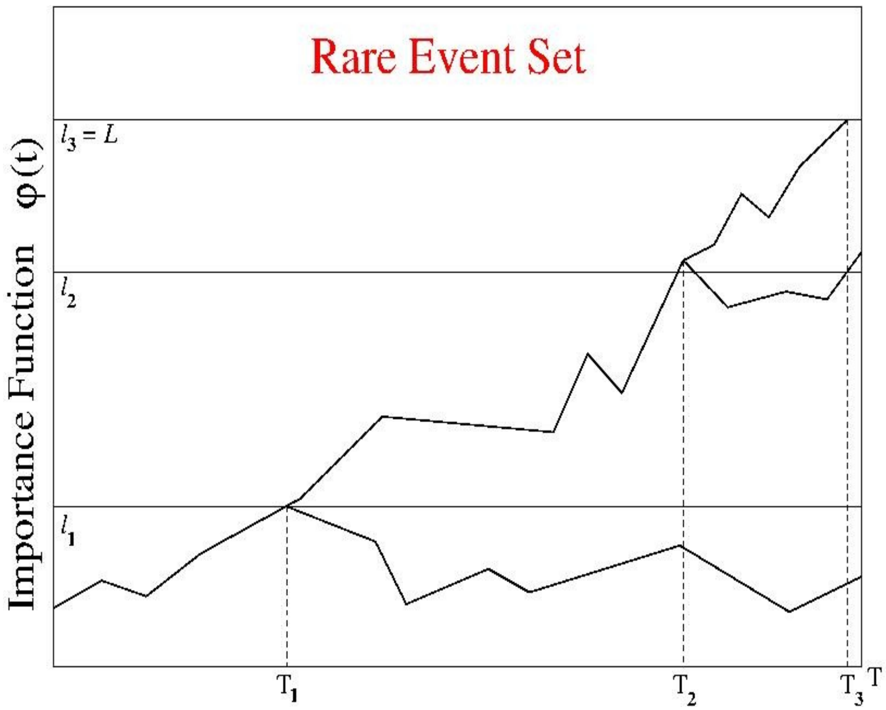
Splitting technique

- *Importance Function* (IF) measures distance to A.

- Decompose distance to A into various '**non-rare**' levels of the IF.

- Sample paths of stochastic processes involved split into multiple copies at various IF levels till A is reached.

- Probability of hitting each **'not-rare'** level



$$p_k = \frac{R_k}{S_{k-1}}.$$

R_k: number of hits at level k

S_k: total number of sample paths launched at level k.

Splitting technique...

- P(A): $\gamma = \prod_{k=1}^{m} p_k$

Splitting technique...

- P(A): $\gamma = \prod_{k=1}^{m} p_k$

- Calculating p_k for each level is not expensive!

Splitting technique...

- P(A): $\gamma = \prod_{k=1}^{m} p_k$

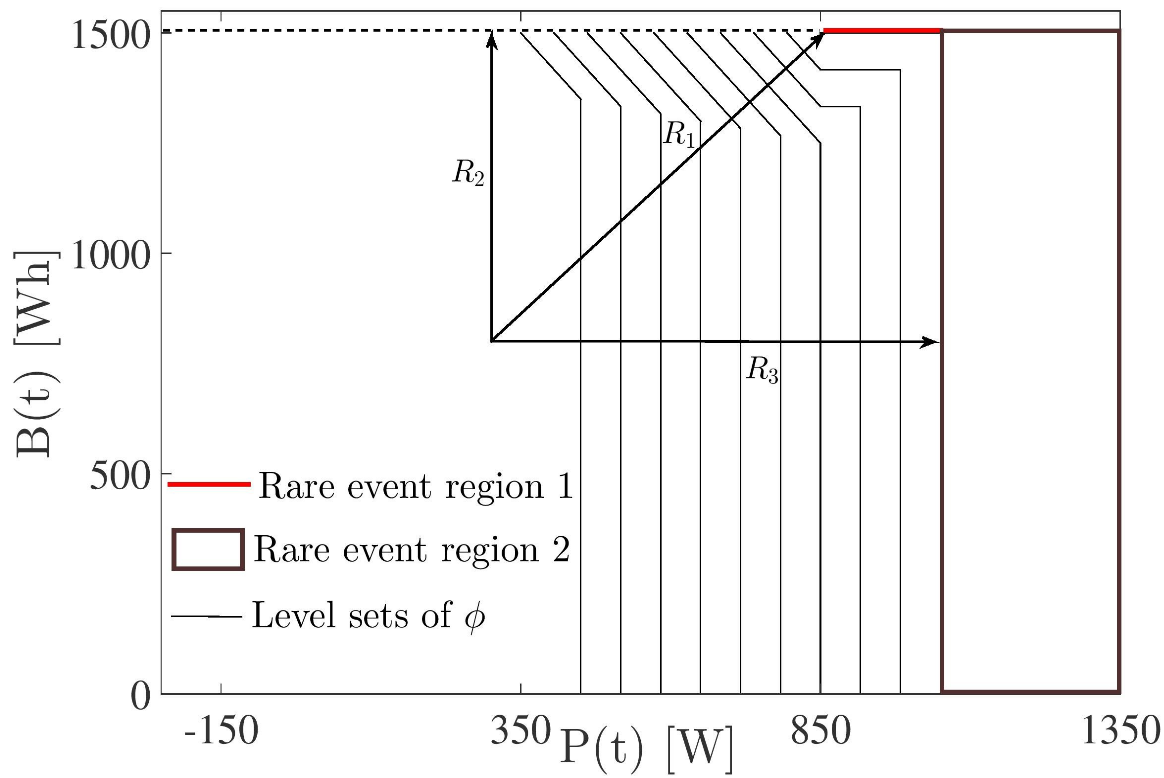
- Calculating p_k for each level is not expensive!

Importance Function is the most important ingredient of splitting.

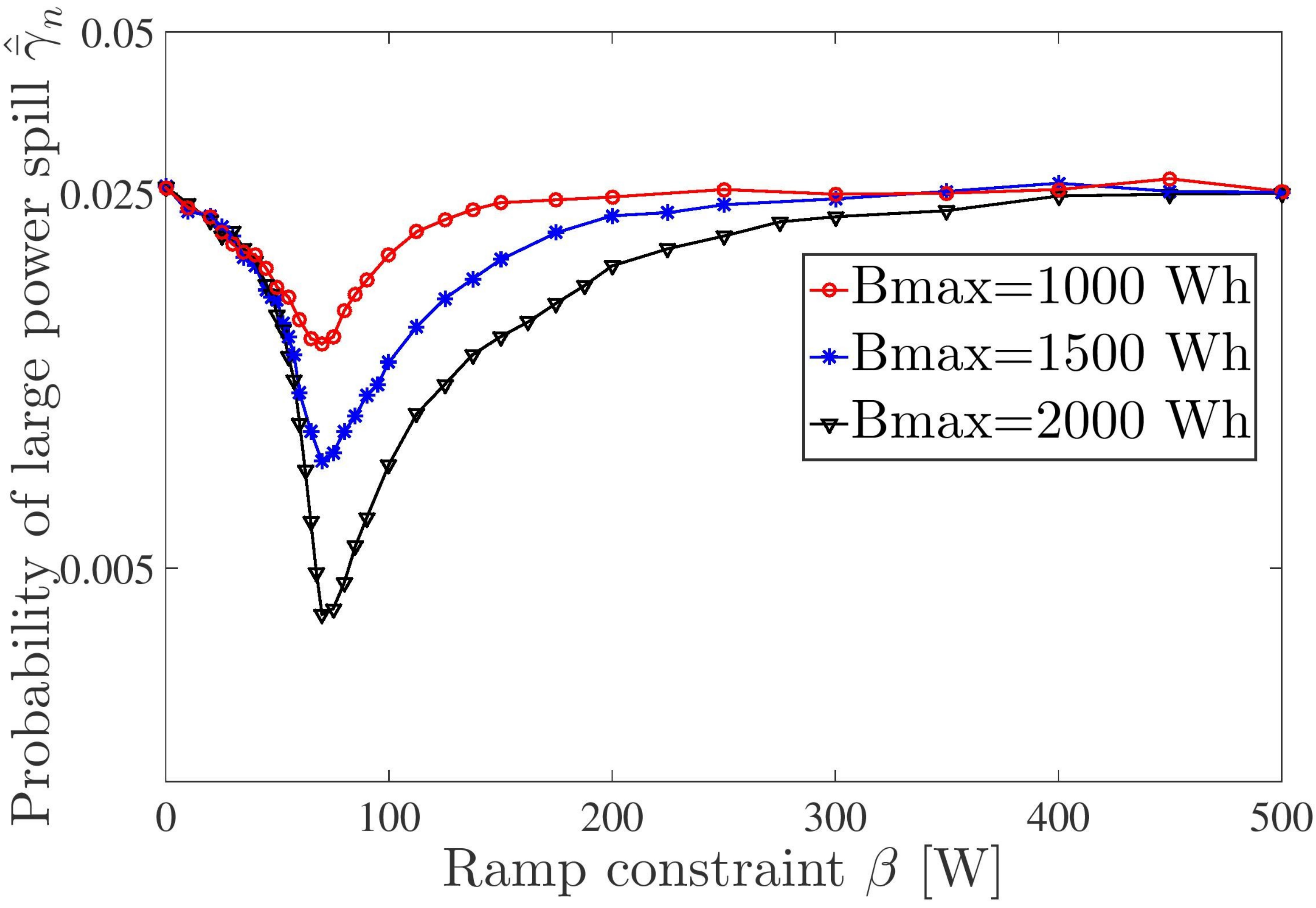
Importance Function for PLPS

We take the IF as the distance from the rare-event sets in the phase space of $B(t)$ and $P(t)$.

$$\varphi(P(t),B(t)) = \begin{cases} -min(R_{1,}R_3) & if \quad P(t) < F_0 \\ -min(R_{2,}R_3) & if \quad P(t) \geqslant F_0 \end{cases} \qquad (4)$$

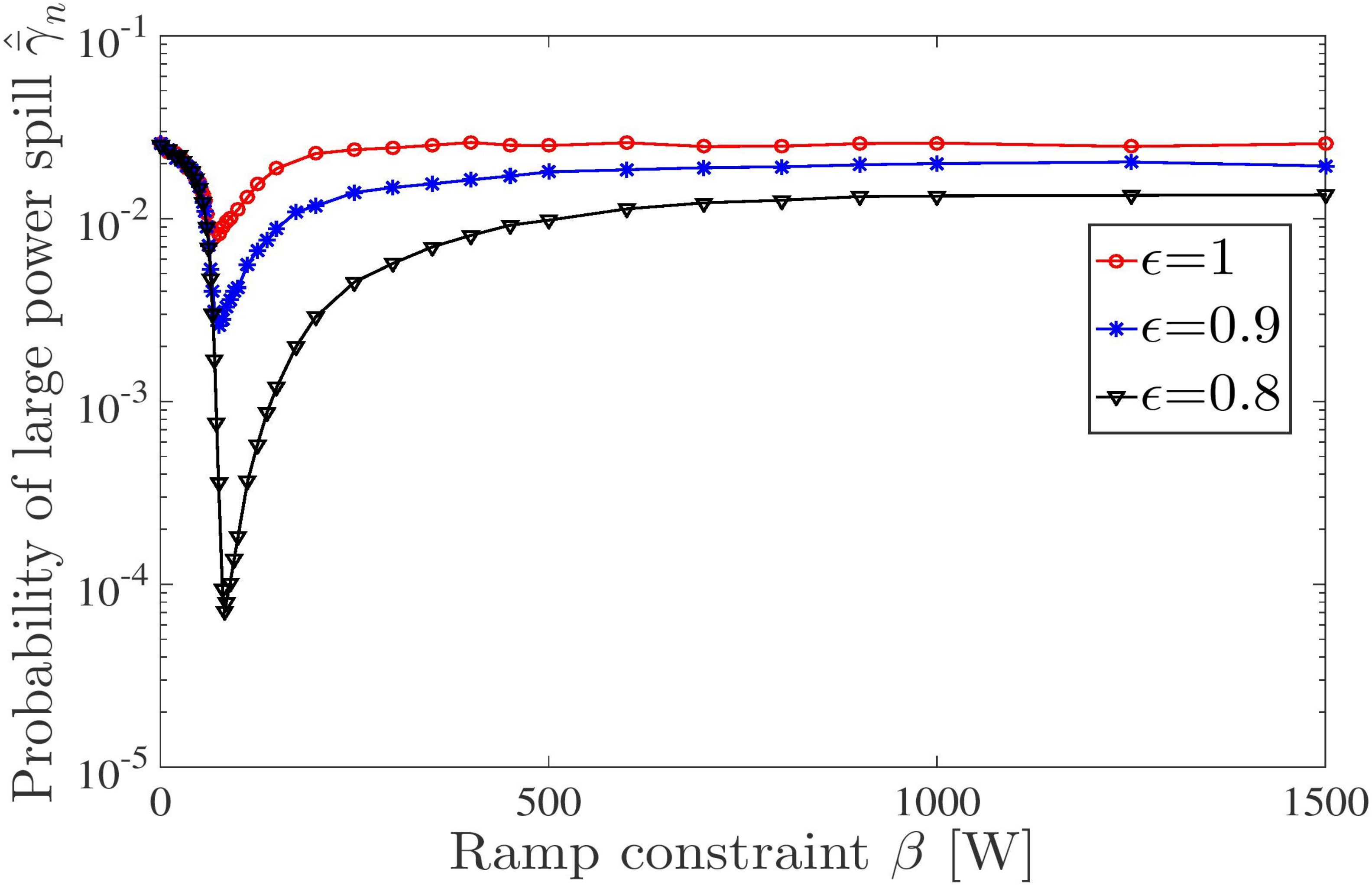


How does the ramp constraint β affect PLPS?



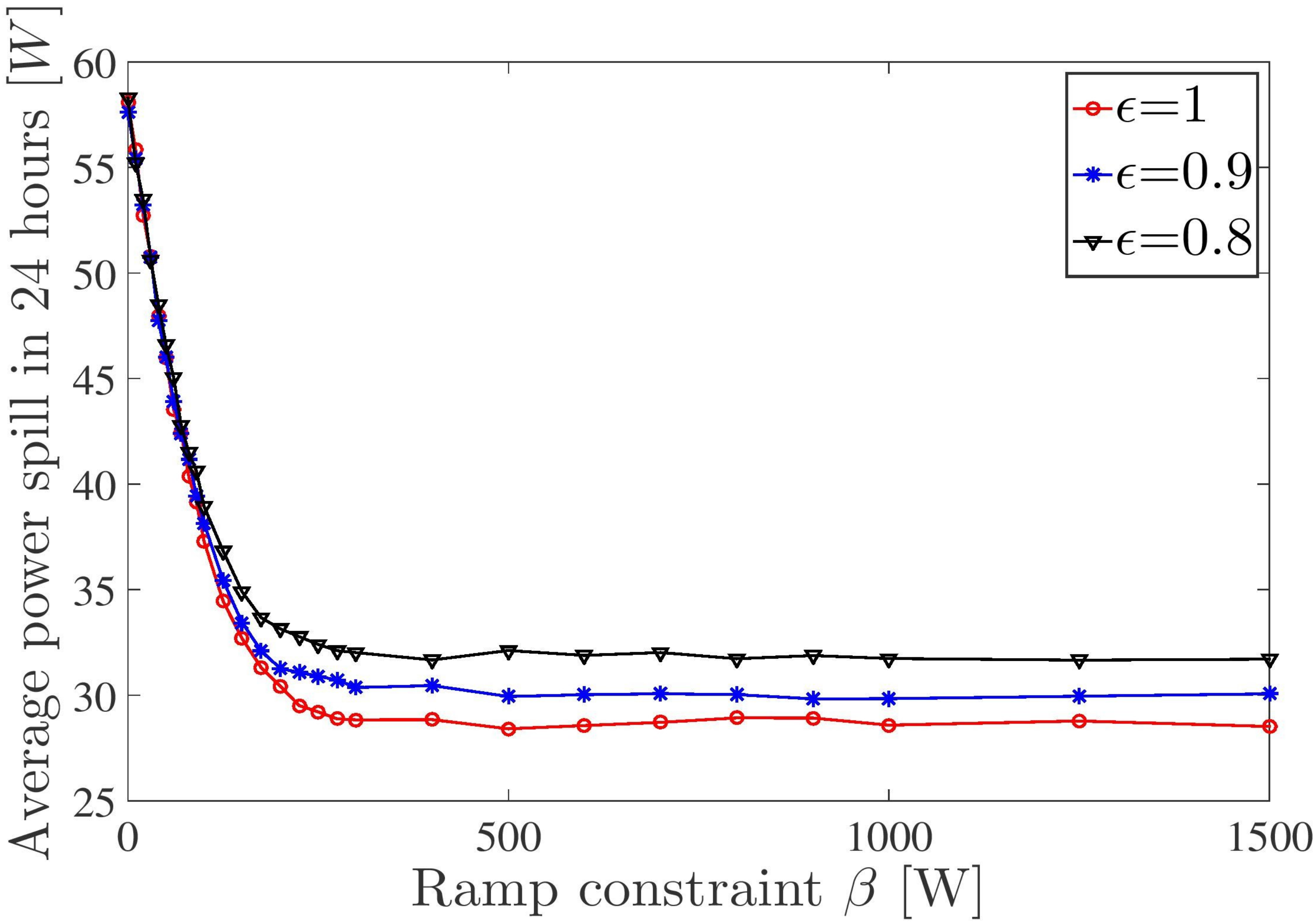
New battery charging strategy to reduce PLPS

A fraction of the battery 1-ε is reserved only for absorbing P(t) ≥ F_0, 0≤ε≤1.



The price of reducing PLPS!

The charging scheme increases the average power spill of the system.



Increase in average power spill is nominal.

The battle of CMC and Splitting

Probability	$\text{CPU-time}_{\textbf{CMC}}/\text{CPU-time}_{\textbf{Splitting}}$
1.8×10^{-2}	8
3.6×10^{-4}	59
7.1×10^{-5}	274

In short

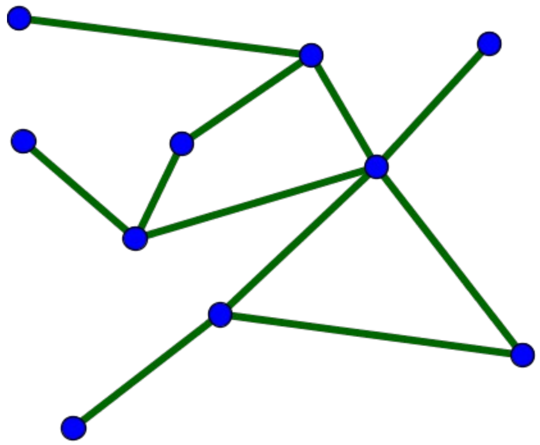
- *Ramp constraints* play a major role is reducing PLPS.
- The charging scheme prescribed reduces PLPS but it comes with a trade-off of increasing the average power spill.
- We find that using splitting over CMC pays off very well.

Optimal Storage Placement in Power Network to Enhance Reliability

What is power network reliability?

Power network is defined as a graph

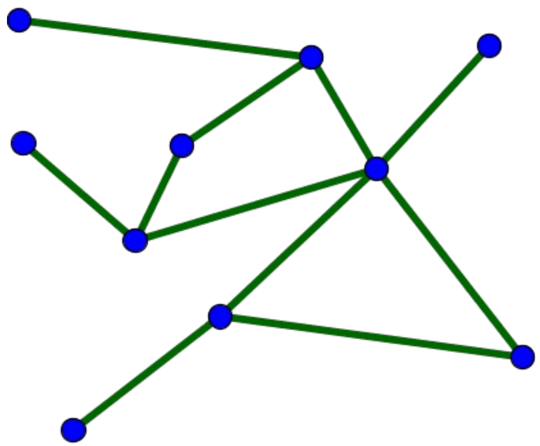
- N nodes
- E edges



What is power network reliability?

Power network is defined as a graph

- N nodes
- E edges



For reliable operation network constraints should not be violated.

1. Voltage constraints : $V^{min} \le |\boldsymbol{V}(t)| \le V^{max} \;\; \forall t \in [0,T], \forall N \quad (1)$

2. Line current constraints : $|\boldsymbol{I}(t)| \le I^{max} \;\; \forall t \in [0,T], \forall E \quad (2)$

How is network reliability challenged?

Same culprits as the last problem!

Energy storage rescues again

Reliability index for DC power flow

Probability of Line Current Violation (PLCV)

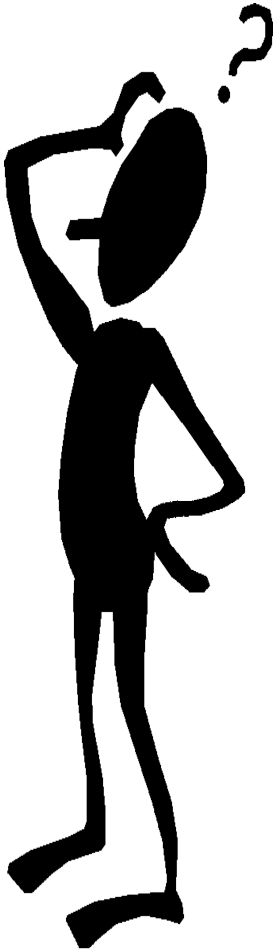
Probability that one of the line currents have exceeded its allowed maximum over period T :

$$\gamma := P\{\ \exists\ (i, j) \in E : sup_{t\in\ [0,T]} \left|I_{(i\ ,\ j)}(t)\right| \geq I_{(i\ ,\ j)}^{max}\ \} \qquad (3)$$

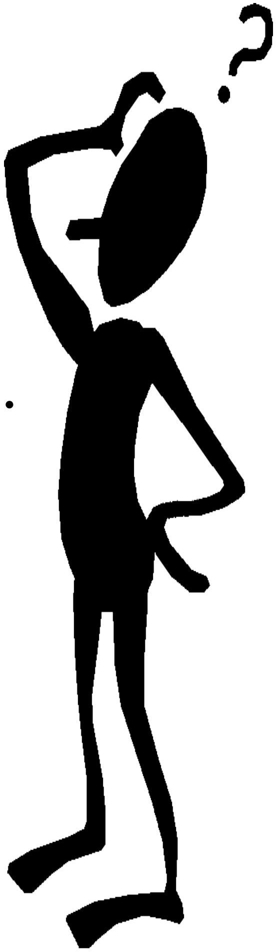
Optimal storage placement problem

Given a network topology and the total storage installation size we wish to optimally place the storage devices in the network such that PLCV is minimal.

Challenges of the problem

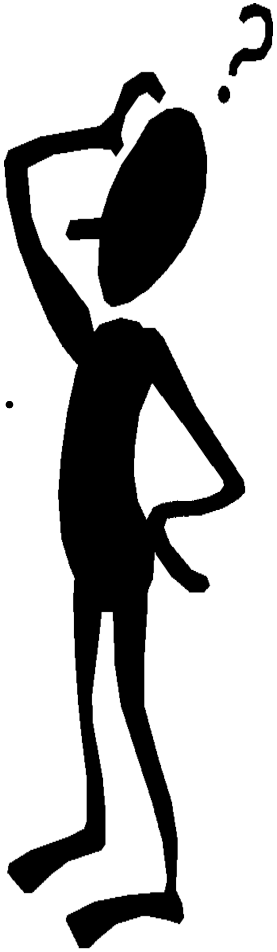


Challenges of the problem



1. Configuration space of storage positions and sizes is very large.

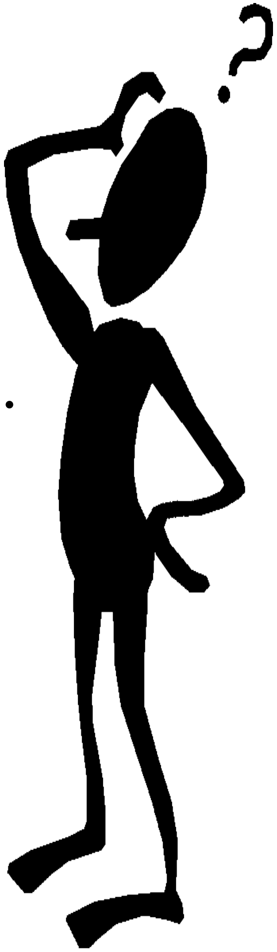
Challenges of the problem



1. Configuration space of storage positions and sizes is very large.

2. We do not expect PLCV to be convex.

Challenges of the problem

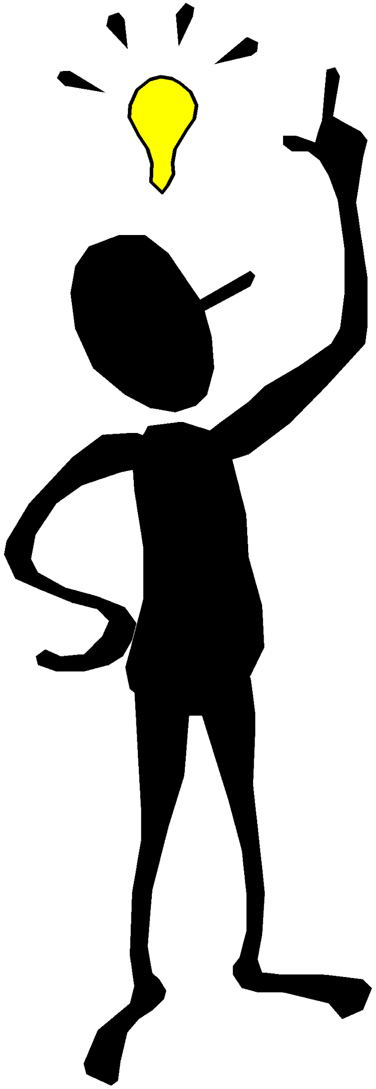


1. Configuration space of storage positions and sizes is very large.

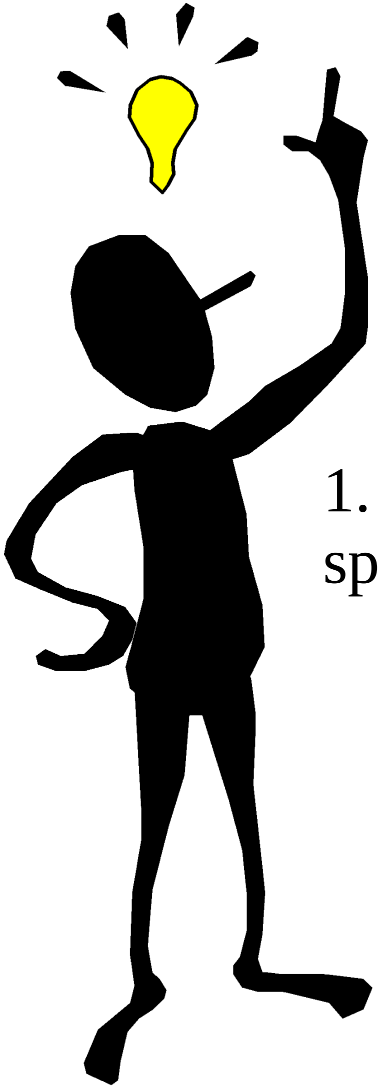
2. We do not expect PLCV to be convex.

3. Calculation of small values of PLCV is expensive using *Crude Monte Carlo* (CMC) simulations.

How to overcome the challenges?

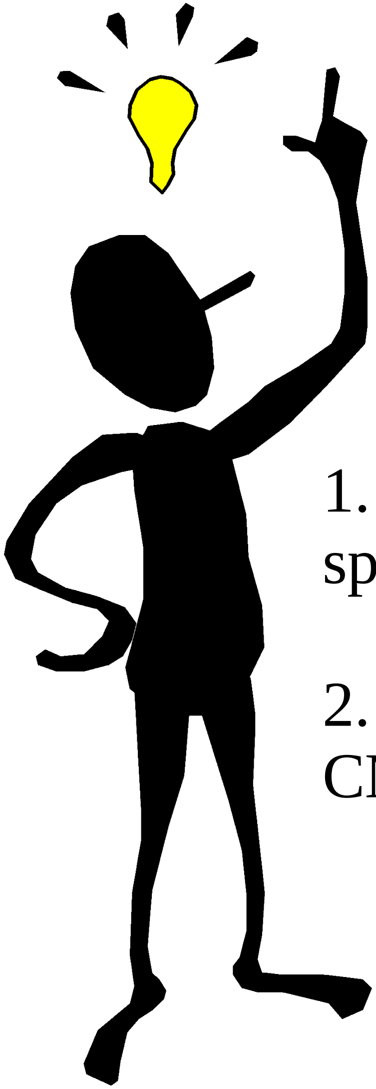


How to overcome the challenges?



1. **Simulated Annealing** (SA) to tackle the large configuration space and non-convexity of the problem.

How to overcome the challenges?



1. **Simulated Annealing** (SA) to tackle the large configuration space and non-convexity of the problem.

2. **Splitting** of rare-event simulations to reduce the workload of CMC.

System setup

Power injection

$P_i(t)$ - net power injection at the *i-th* node modeled as *Ornstein-Uhlenbeck* processes :

$$P_i(t+\Delta t) = P_i(t) + \theta_i[\mu_i - P_i(t)]\,\Delta t + \sigma_i \Delta W_i(t) \quad \forall t \in [0,T] \tag{4}$$

System setup

Storage (battery) model

The batteries are charged locally at each node according to:

$$B_i(t+\Delta t)=B_i(t)+p_i^B(t)\Delta t \qquad (5)$$

$p_i^B(t)$: power flowing in/out of the *i-th* battery.

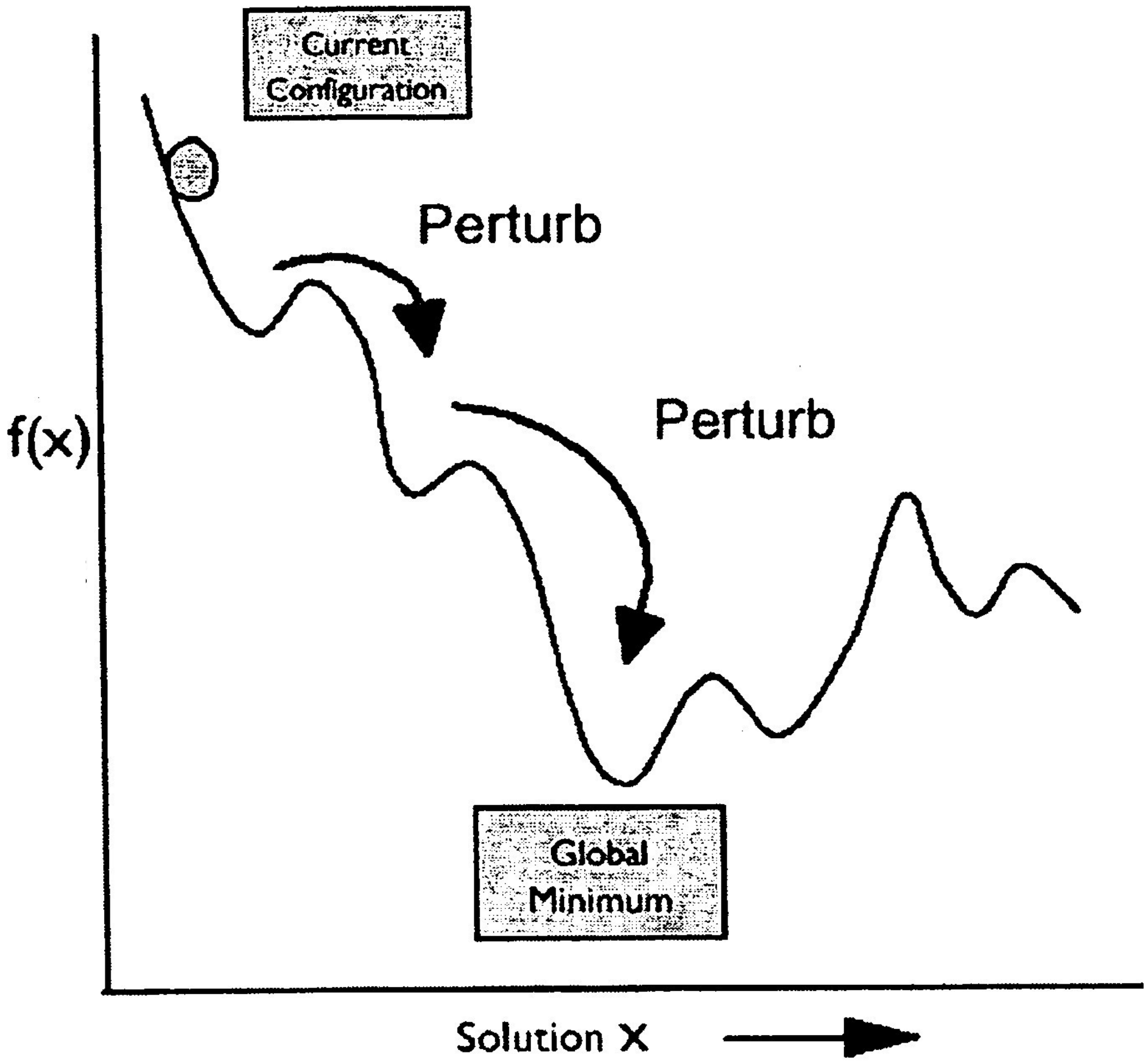
The batteries are bounded by

1. *capacity constraints* : $0 \le B_i(t) \le B_i^{max}$ (6)

2. *total installation constraint* : $\sum_{i=1}^{N-1} B_i^{max} = B^{max}$ (7)

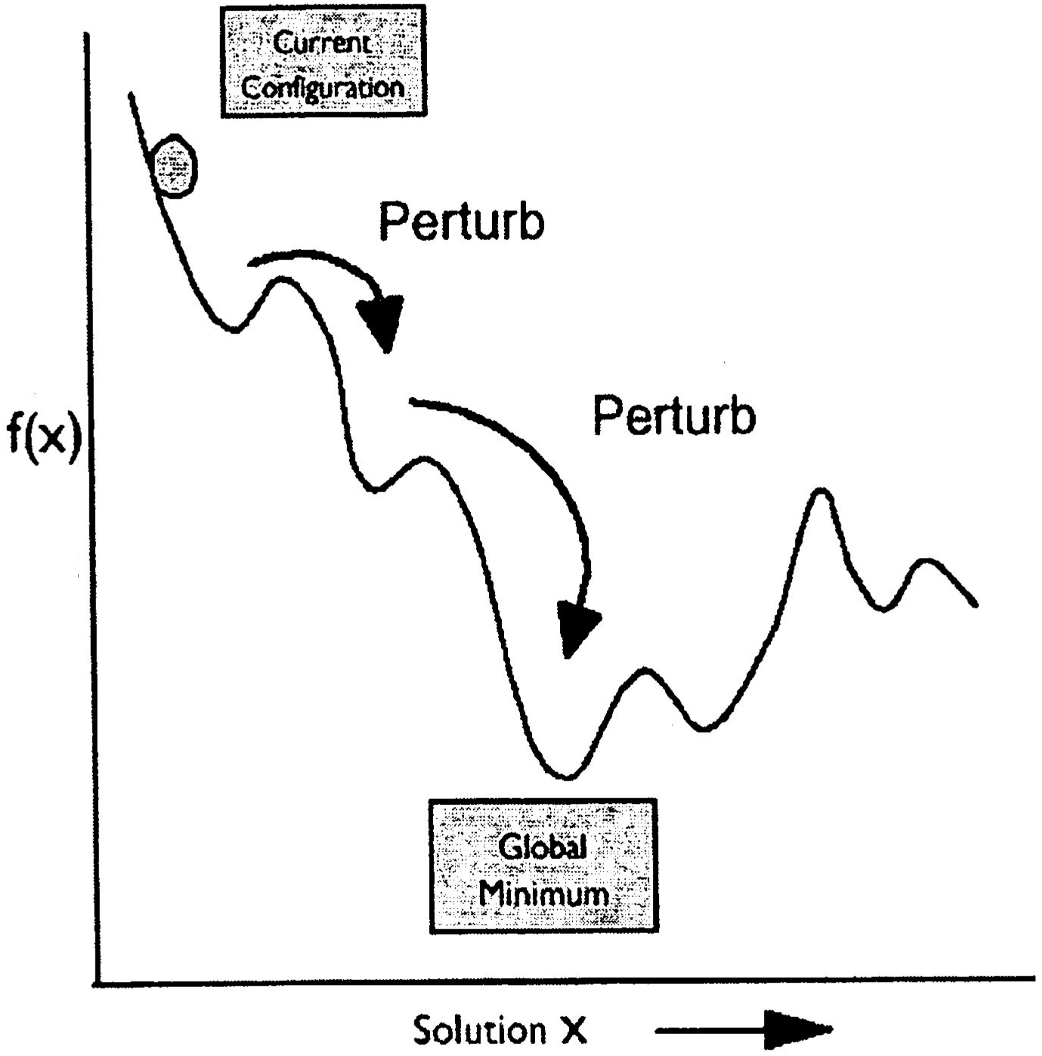
Simulated Annealing (SA)

Performs a local search in the solution space ***X*** of the problem to minimize or maximize a desired ***cost function*** $f(\boldsymbol{X})$.



Simulated Annealing (SA)

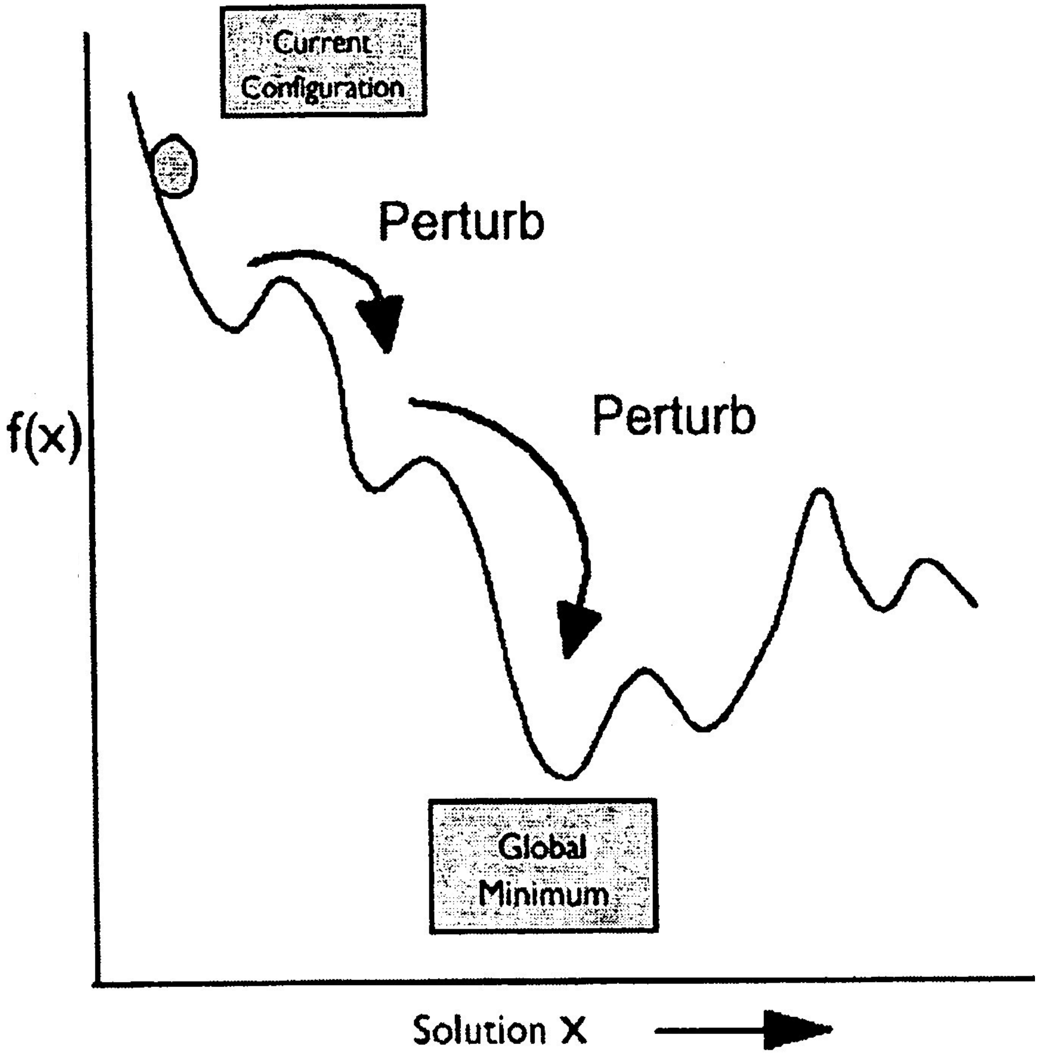
Performs a local search in the solution space ***X*** of the problem to minimize or maximize a desired ***cost function*** *f*(***X***).



Our Problem

Simulated Annealing (SA)

Performs a local search in the solution space ***X*** of the problem to minimize or maximize a desired ***cost function*** $f(\boldsymbol{X})$.

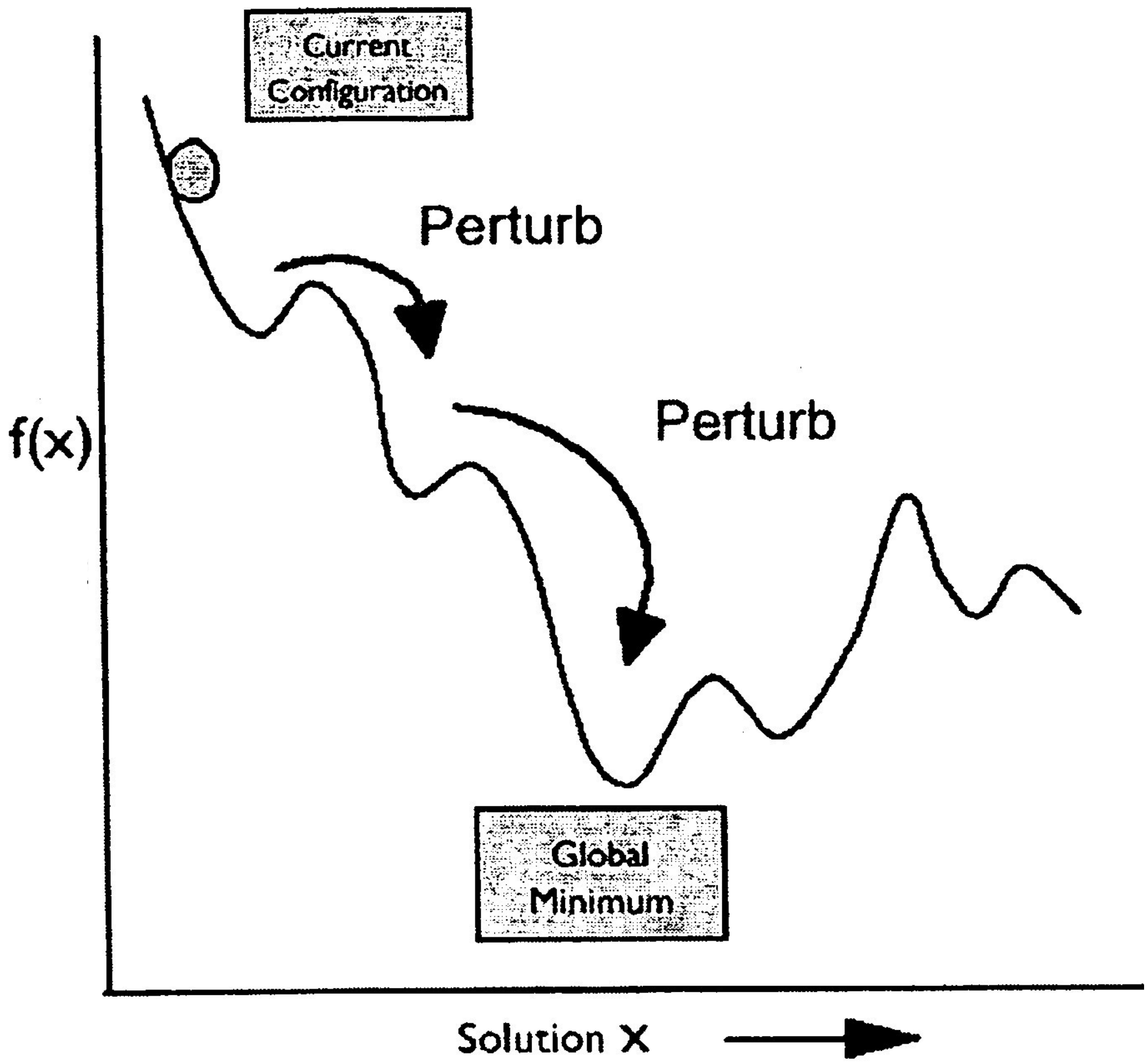


Our Problem

- Minimize $f(\boldsymbol{X})$: γ
- Solution space ***X*** : B_i^{max}

SA Algorithm

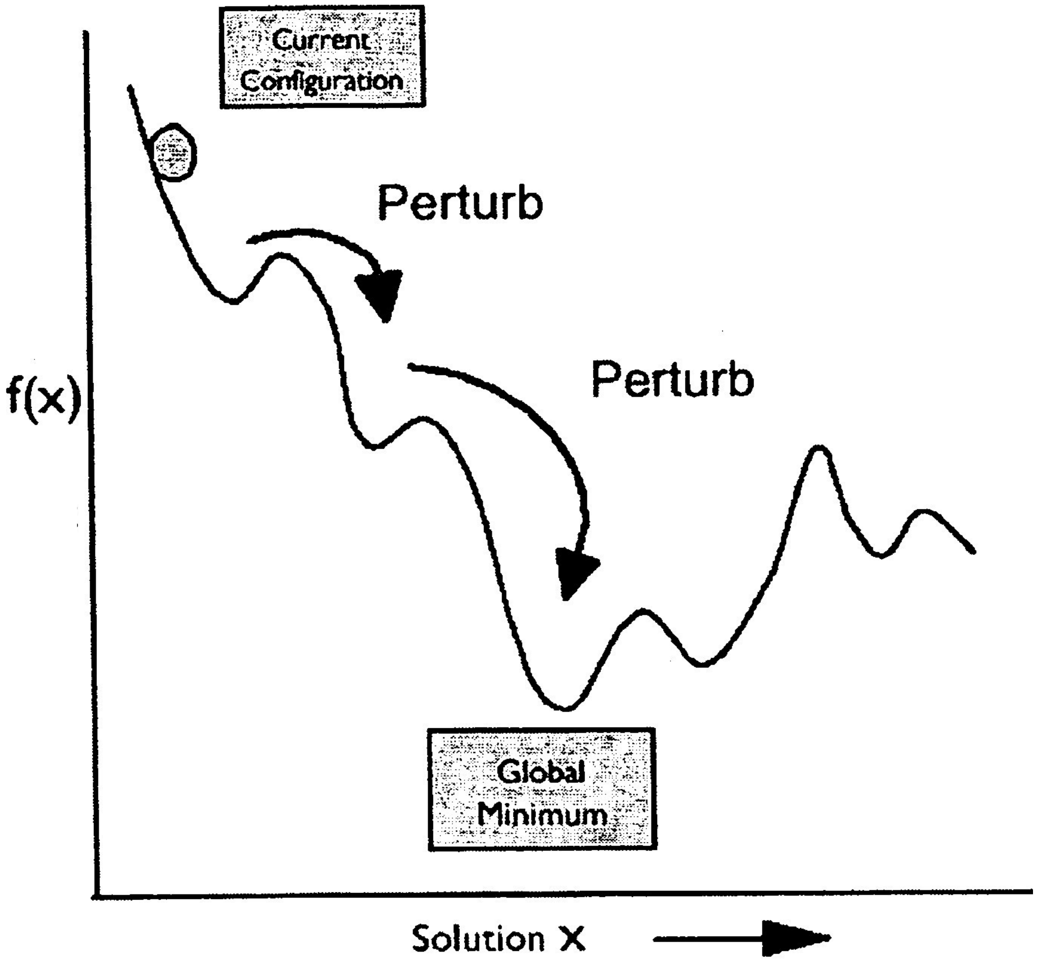
1. Start with an initial solution as $\boldsymbol{X}^{best}$. Initialize $T_{c.}$



SA Algorithm

1. Start with an initial solution as X^{best} . Initialize $T_{c.}$

2. Randomly select a new solution X^* in the solution configuration space.



SA Algorithm

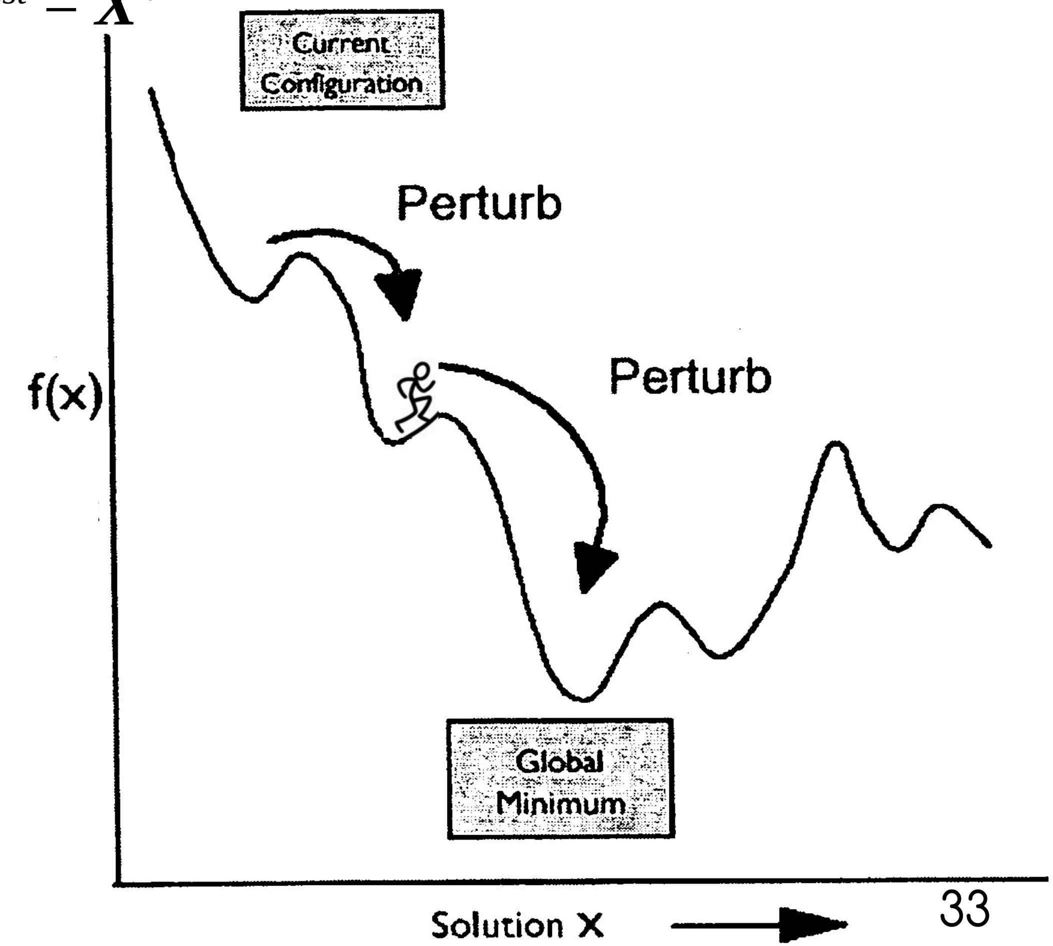
1. Start with an initial solution as $\boldsymbol{X}^{best}$. Initialize $T_{c.}$
2. Randomly select a new solution $\boldsymbol{X}^*$ in the solution configuration space .

3. If $\Delta E = f(\boldsymbol{X}^*) - f(\boldsymbol{X}^{best}) < 0$, then $\boldsymbol{X}^{best} = \boldsymbol{X}^*$

   ElseIf $\Delta E = f(\boldsymbol{X}^*) - f(\boldsymbol{X}^{best}) > 0$,
   accept the *worse* solution as $\boldsymbol{X}^{best}$
   with *acceptance probability*,

   $$p = exp(-\Delta E/T_c).$$

   Helps escape *local minima*!



SA Algorithm...

1. Start with an initial solution as $\boldsymbol{X}^{best}$. Initialize $T_{c.}$

2. Randomly select a new solution $\boldsymbol{X}^*$ in the solution configuration space .

3. If $\Delta E = f(\boldsymbol{X}^*) - f(\boldsymbol{X}^{best}) < 0$, then $\boldsymbol{X}^{best} = \boldsymbol{X}^*$.

   ElseIf $\Delta E = f(\boldsymbol{X}^*) - f(\boldsymbol{X}^{best}) > 0$, then accept the new *worse* solution as the best solution with *acceptance probability*,
   $$p = exp(-\Delta E/T_c).$$
   This helps the algorithm to escape *local minima*.

4. Cool the temperature of the probability of accepting *worse* solutions p , $T_c^{new} = \kappa T_c^{old}$, where $0<\kappa<1$.

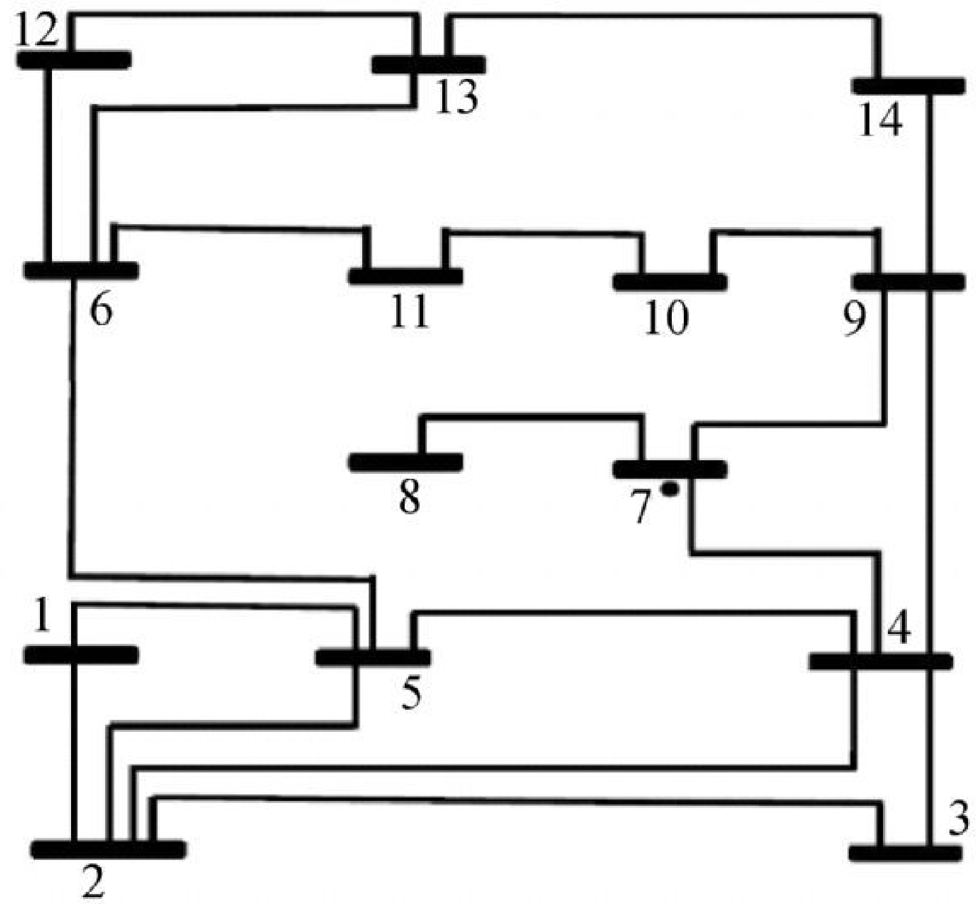
5. Repeat 2 until the *stopping criterion* is reached.

Splitting technique

Importance Function for PLCV

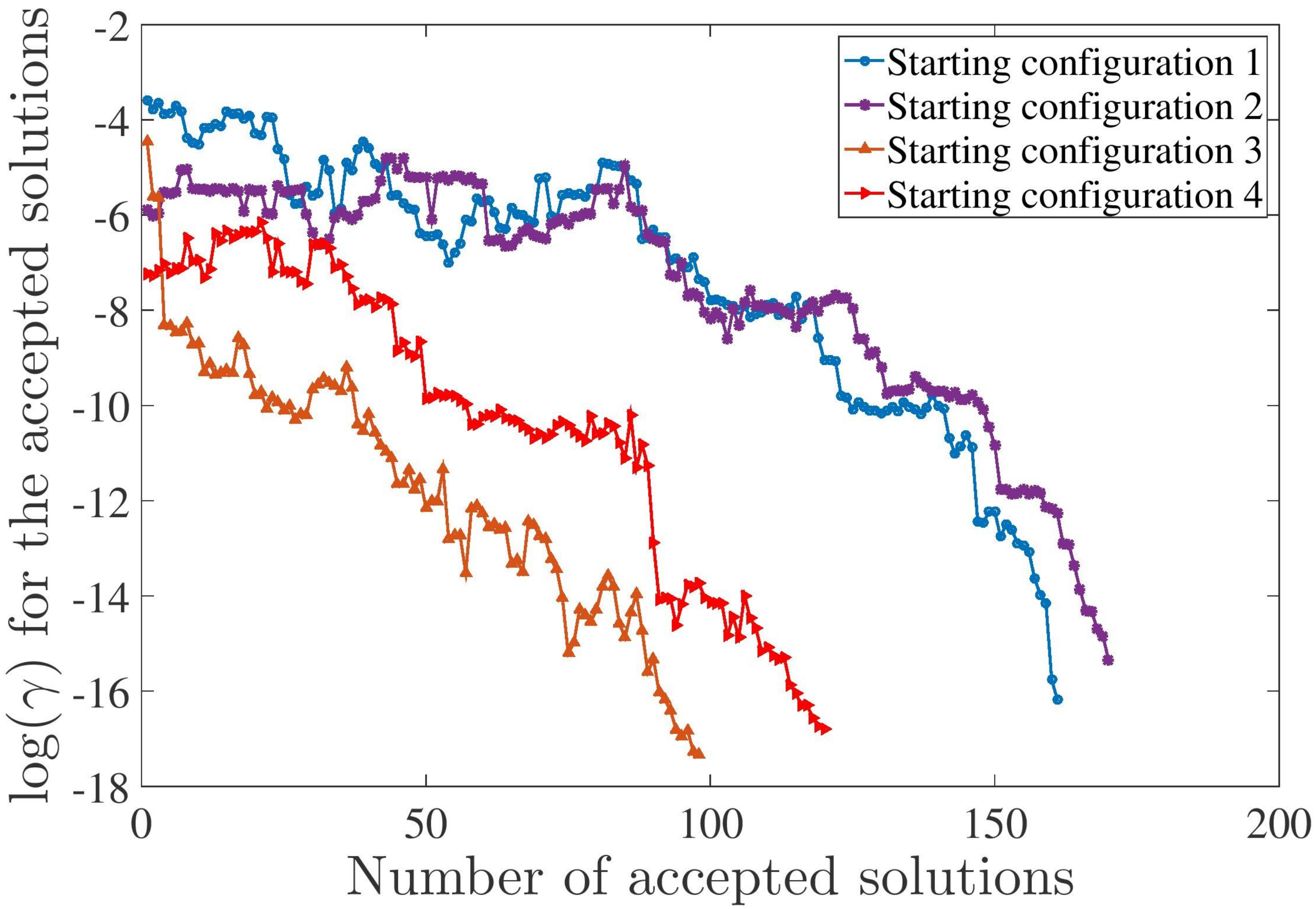
$$\varphi(|I_{(i,j)}(t)|) = \max_{(i,j)\in E} \frac{|I_{(i,j)}(t)|}{I_{(i,j)}^{max}} \qquad (8)$$

IEEE 14 bus

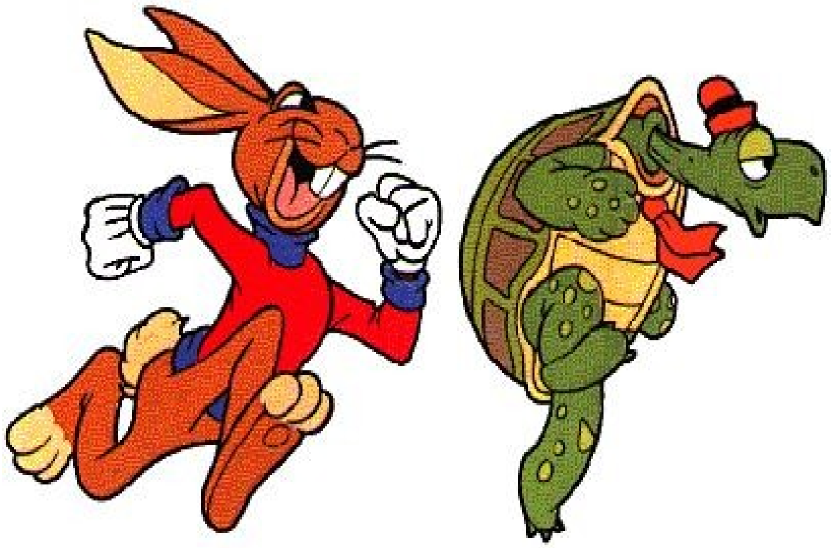


Different initial configurations

- We start from different initial configurations of the battery positions and sizes to minimize log(γ).
- B^{max} = 13000 units.



CMC versus Splitting



Probability	$\text{CPU-time}_{\mathbf{CMC}}/\text{CPU-time}_{\mathbf{Splitting}}$
1.0×10^{-2}	4
1.25×10^{-3}	20
1.0×10^{-4}	80

In short

- Using SA we minimized γ from ~ 10^{-3} to 10^{-7}.
- We find that using splitting over CMC pays off very well.
- For more : http://oai.cwi.nl/oai/asset/23857/23857A.pdf

Cost function for the problem : log(γ) not γ!

- While minimizing γ values can go down to ~ 10^{-5}−10^{-7} or smaller depending on the total installation size of the battery.

- The *acceptance probability p* of the *worse* solution also depends on $\Delta E = \gamma(\boldsymbol{X}^{*}) - \gamma(\boldsymbol{X}^{best})$.

- As the γ's are very small, their differences are also small hence *p* becomes large and the algorithm accepts too many *worse* solutions and might never converge.

- So, instead of minimizing γ we minimize log(γ) so that $\Delta E = \log(\gamma(\boldsymbol{X}^{*})) - \log(\gamma(\boldsymbol{X}^{best}))$ is not very small and the algorithm does not accept too many *worse* solutions.

Perturbations in solution space

Solution space

The solution space of the system is the configuration space of battery positions and sizes at each node in the network.

Perturbations

- SA searches in the solution space by randomly perturbing the system.
- To move randomly in the solution configuration space of the battery positions and sizes we randomly select two non-slack nodes $(i, j)\ \forall\ i \in N/\{1\}$ and $\forall\ j(\neq i) \in N/\{1\}$.
- Then exchange m units of battery blocks ΔB between the two chosen nodes such that the following are true :
  1. The total installation storage capacity remains constant

$$\sum_{i=1}^{N-1} B_i^{max} = B^{max}$$

2. For $i = 1, \ldots, N-1$: $0 \leq B_i^{max} \leq B^{max}$