On the Convergence Time of Simulated Annealing

Siyeong Lee

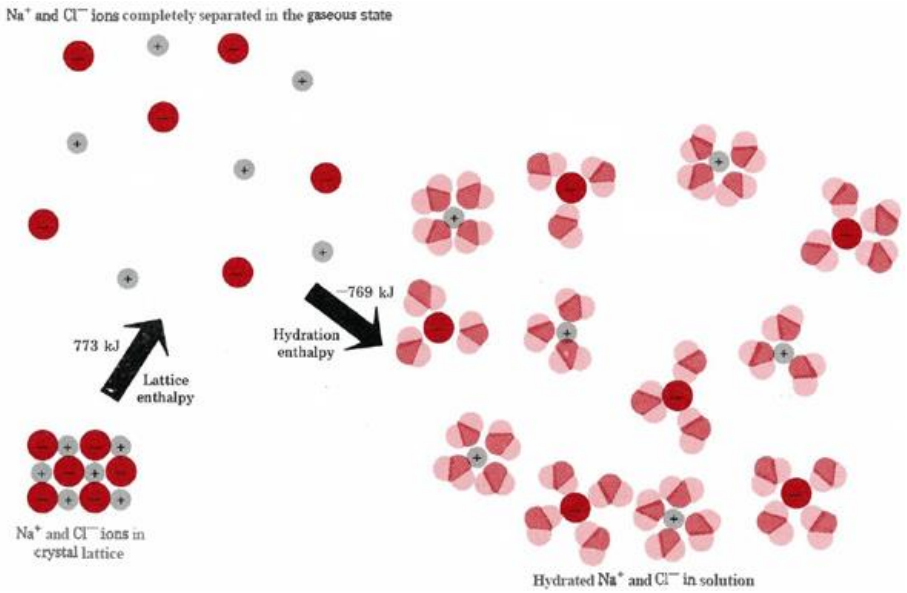
Dept. of Electronic Engineering

Sogang University

May 9, 2017

Introduction

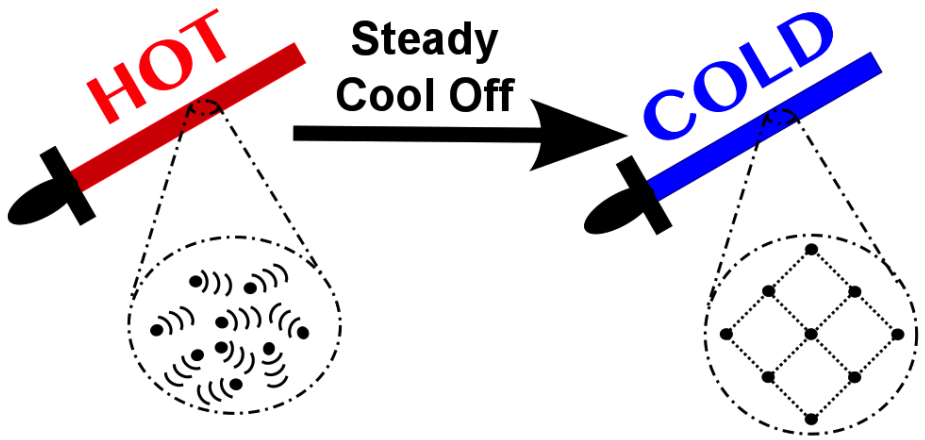
- Annealing
  - A crystalline solid is heated and then allowed to cool very slowly until it achieves its most regular possible crystal lattice configuration (i.e., its minimum lattice energy state).

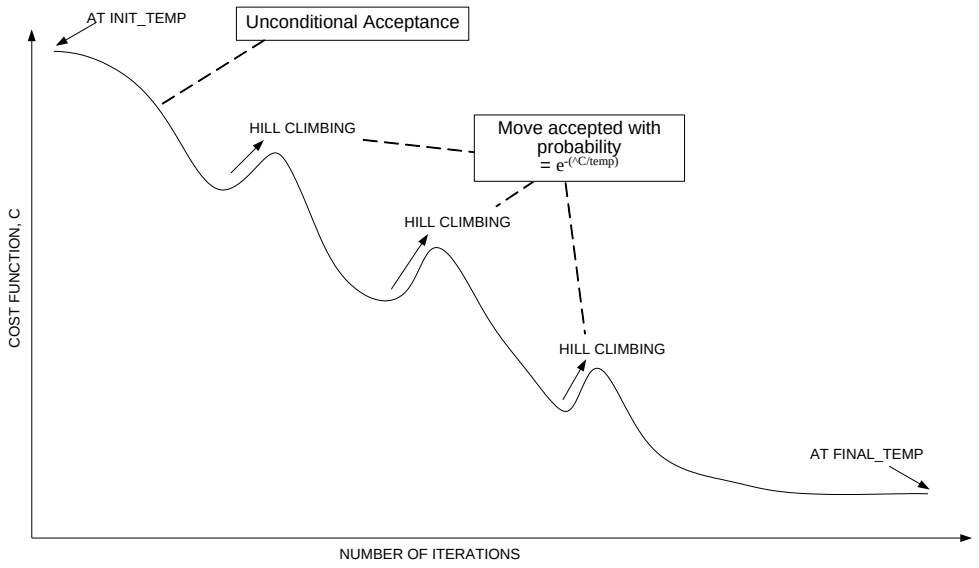


Introduction

- Similarity between Annealing and Combinatorial Opt. problem
  - High temperature → Cooling slowly; **Free energy ↓**
  - Finding the optimal solution ; **Cost function value ↓**

Solving strategy: *Simulated Annealing*





Problem setting – Optimization problem

Definition 1. Optimization problem

X: *the search set*

$$Minimize\ f_0(x)$$

$$Subject\ to\ f_i(x) \leq b_i, i = 1, \dots, m$$

$f_0 : \boldsymbol{R}^n \to \boldsymbol{R}$; Objective function
$f_i : \boldsymbol{R}^n \to \boldsymbol{R}, i = 1, \dots m$: Constraint functions

$x = (x_1, \dots, x_n) \in \boldsymbol{X}$: *Optimization variables*

Specially, solutions which satisfy all constraints is called ***feasible solutions***

Relationship between Physical Annealing and Simulated Annealing

Thermodynamic Simulation	Combinatorial Optimization
System states	Feasible Solutions
Energy	Cost
Change of State	Neighbouring Solutions
Temperature	Control Parameter T
Frozen State	Heuristic Solution
Ground State	Optimal Solution

Rapid Quenching	**Local Search**
Careful Annealing	Simulated Annealing

Representation

- Teminology
  - Given a cost function $C(p_1, p_2, \cdots, p_n)$
    - a **configuration** or **state** of the Opt. problem is defined to be an assignment of values to its n parameters.
    - **'Neighbors'** of given state are all those states which differ from the given state only in the value of one parameter by a 'small' amount
    - **'Temperature'** is simply a control parameter which has the same units as that of the cost function.

Pseudocode

```
procedure Anneal(start_state, start_temperature);

    while (not frozen) do
        while (not steadystate) do
            Generate a random neighbor of the current state;
            Let Δ = cost of old state − cost of the new state;
            Accept the new state with probability min{1, exp(−Δ/T)},
            T being the current temperature;
        Update the temperature; steadystate = false;
```

Frozen

becomes true
when no improvement in the given cost function has been observed long time.

Steady State

becomes true
when the system attains steady state at the given temperature.

e.g) L time Loop

Pseudocode

```
procedure Anneal(start_state, start_temperature);
    while (not frozen) do
        while (not steadystate) do
            Generate a random neighbor of the current state;
            Let Δ = Cost of new state − Cost of old state
            Accept the new state with probability min{1, exp(−Δ/T)},
            T being the current temperature;
        Update the temperature; steadystate = false;
```

Frozen

becomes true
when no improvement in the given cost function has been observed long time.

Steady State

becomes true
when the system attains steady state at the given temperature.

e.g) L time Loop

Pseudocode

```
procedure Anneal(start_state, start_temperature);

    while (not frozen) do

        while (not steadystate) do

            Generate a random neighbor of the current state;
            Let Δ = Cost of new state − Cost of old state
            Accept the new state with probability min{1, exp(−Δ/T)},
            T being the current temperature;

        Update the temperature; steadystate = false;
```

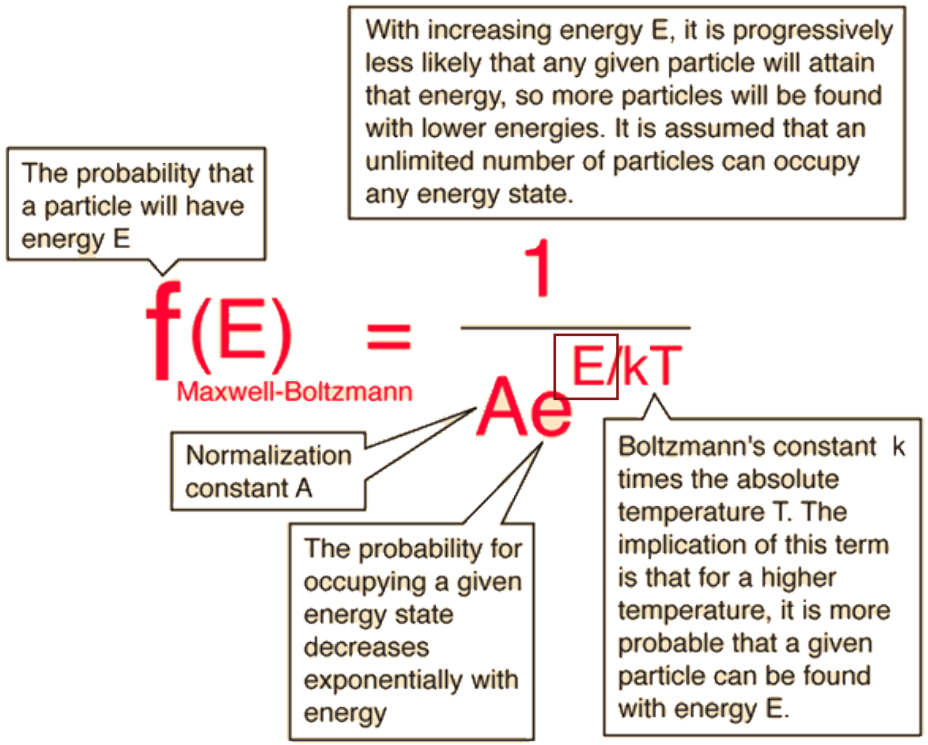
1. Get an initial solution S.
2. Get an initial temperature $T > 0$.
3. While not yet *frozen* do the following.
   - 3.1 Perform the following loop L times.
     - 3.1.1 Pick a random neighbor S' of S.
     - 3.1.2 Let $\Delta = cost\ (S') - cost\ (S)$.
     - 3.1.3 If $\Delta \leqslant 0$ (downhill move), Set $S = S'$.
     - 3.1.4 If $\Delta > 0$ (uphill move), Set $S = S'$ with probability $e^{-\Delta/T}$.
   - 3.2 Set $T = rT$ (reduce temperature).
4. Return S.

Formulating SA

Real Annealing

$given\ T, \min\left(1, e^{-\frac{\Delta}{T}}\right)$

- Boltzmann distribution



Simulated Annealing

- Suppose that
  $\Delta = C(new\ state) - C(old\ state) < 0$
  - Since $e^{-\frac{\Delta}{T}} > 1$,
    Accept the new state
    with probability $\min(1, e^{-\frac{\Delta}{T}}) = 1$
  - Move!!
- Suppose that
  $\Delta = C(new\ state) - C(old\ state) > 0$
  - Since $e^{-\frac{\Delta}{T}} < 1$,
    Accept the new state
    with probability $\min(1, e^{-\frac{\Delta}{T}}) < 1$
  - maybe move!!

Formulating SA

Real Annealing

$given\ T, \min\left(1, e^{-\frac{\Delta}{T}}\right)$

- Boltzmann distribution

With increasing energy E, it is progressively less likely that any given particle will attain

Maxwell-Boltzmann

$Ae^{-E/kT}$

Normalization constant A

The probability for occupying a given energy state decreases exponentially with energy

Boltzmann's constant k times the absolute temperature T. The implication of this term is that for a higher temperature, it is more probable that a given particle can be found with energy E.

Simulated Annealing

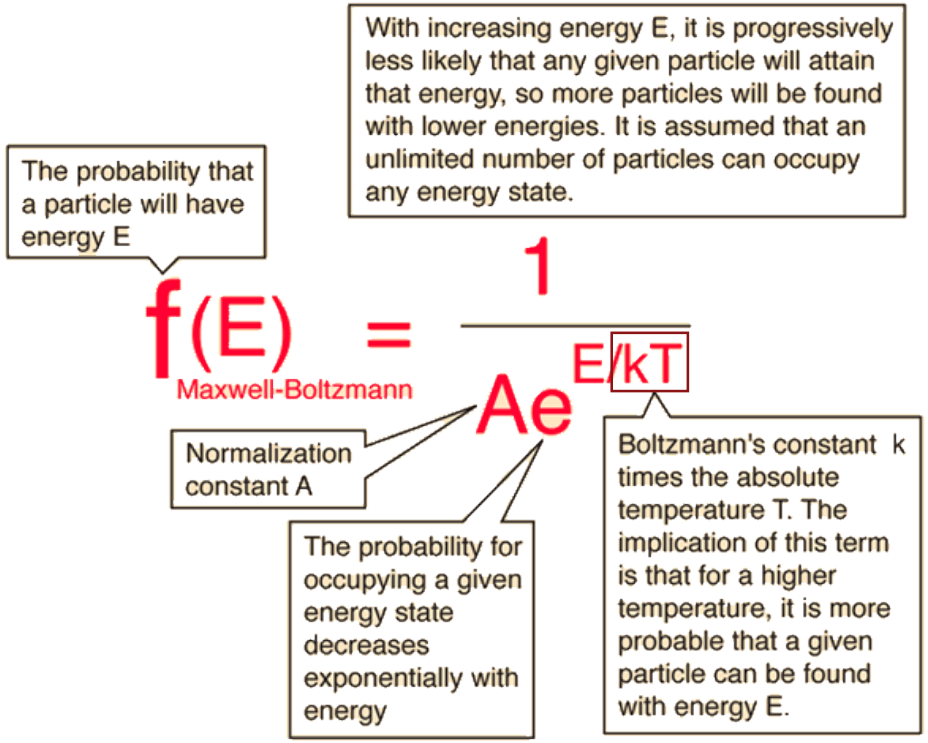
- Suppose that
  $\Delta = C(new\ state) - C(old\ state) < 0$
  - Since $e^{-\frac{\Delta}{T}} > 1$,

Accepting worse solutions is a fundamental property of metaheuristics because it allows for a more extensive search for the optimal solution.

- Suppose that
  $\Delta = C(new\ state) - C(old\ state) > 0$
  - Since $e^{-\frac{\Delta}{T}} < 1$,
    Accept the new state
    with probability $\min(1, e^{-\frac{\Delta}{T}}) < 1$
  - maybe move!!

Formulating SA

Real Annealing



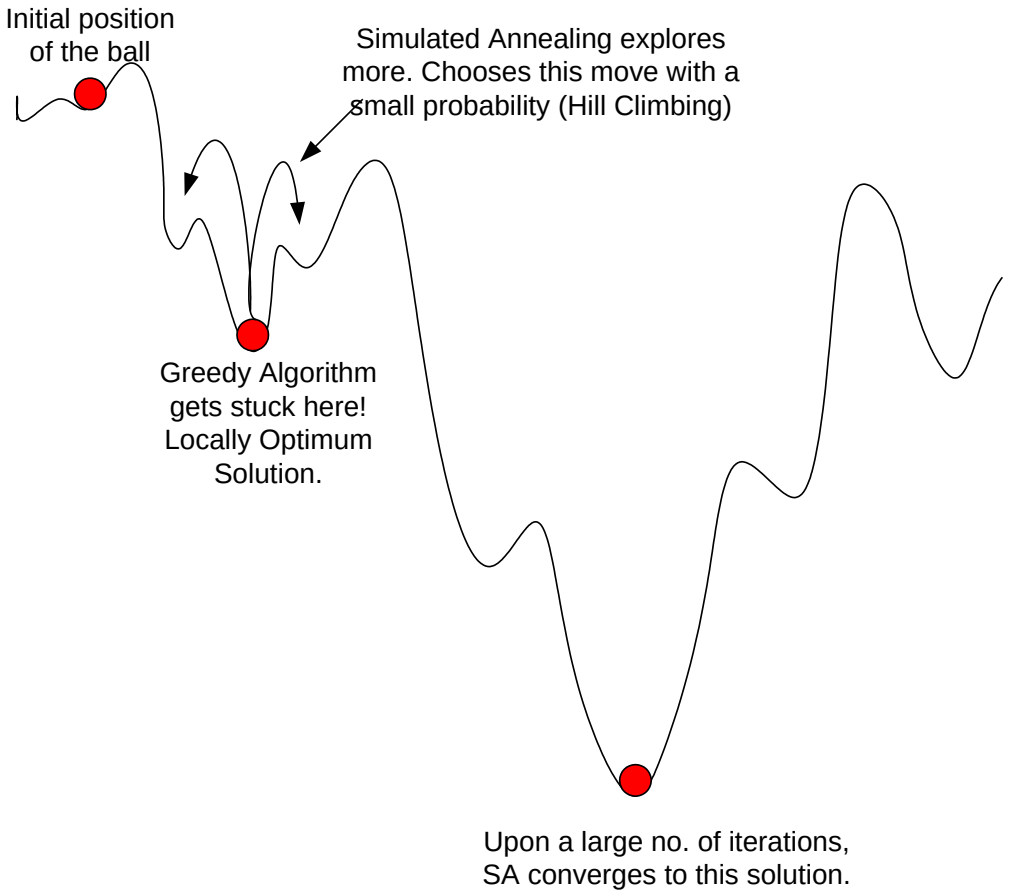
Acceptance Function

- At high temperatures,
  - the probability of accepting worse moves is high.
  - If $T = \infty$, all moves are accepted
  - It corresponds to a random local walk in the landscape.
- At low temperatures,
  - The probability of accepting worse moves decreases.
  - If $T = 0$, no worse moves are accepted
  - The search is equivalent to local search.

Toy problem

Ball on terrain

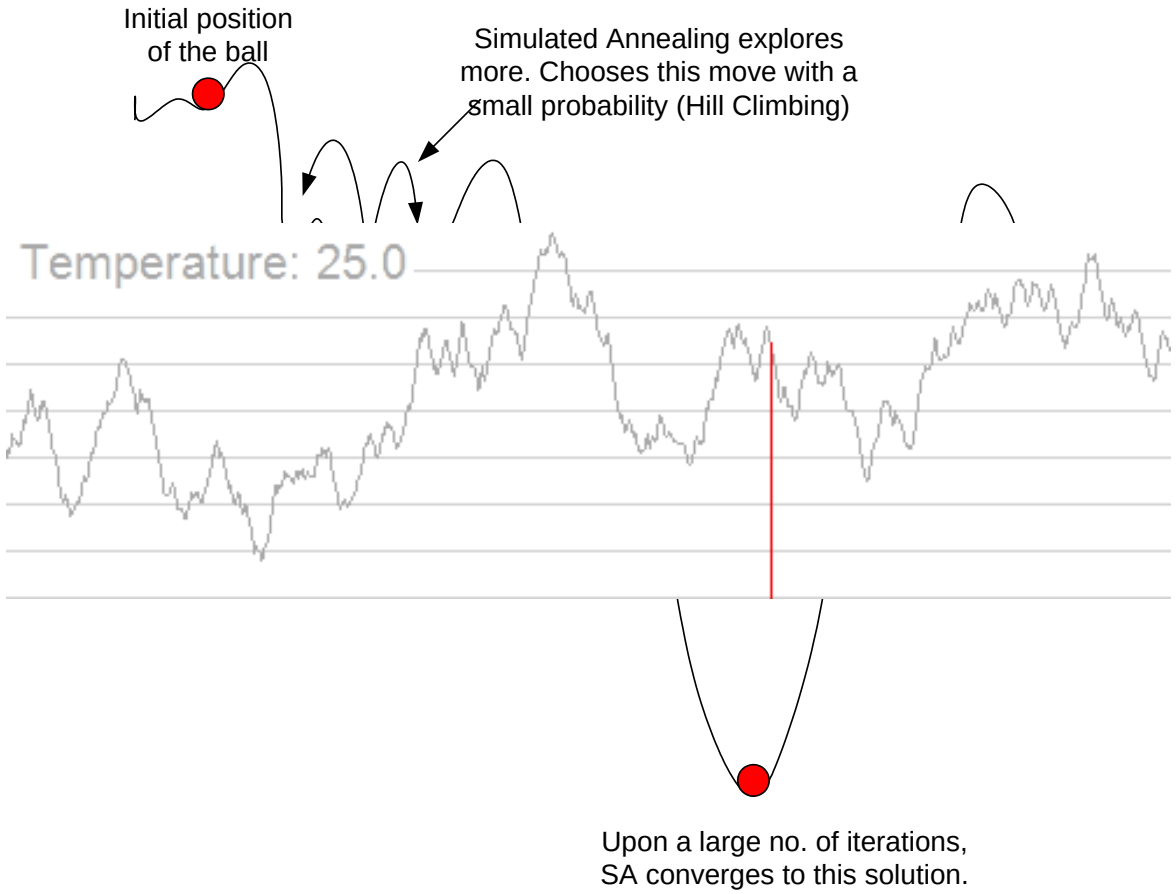
- The ball is initially placed at a random position on the terrain.
- Representation
  - $State$: $position\ (x)$
    $\rightarrow$ *Expressed as a binary number*
  - $Neighboer$
    - *Given x, only single-bit mutation.*
  - $Cost \rightarrow Height$



Toy problem

Ball on terrain

- The ball is initially placed at a random position on the terrain.
- Representation
  - $State$: $position\ (x)$
    $\rightarrow$ *Expressed as a binary number*
  - $Neighboer$
    - *Given x, only single-bit mutation.*
  - $Cost \rightarrow Height$



Markov property in SA

- We can construct a directed graph $G(V, E)$ corresponding to a given OP in the following way.
  - V is a set of states of the OP.
  - Edges going out of any node will be the neighbors of this state.
- SA algorithm performs a random walk on this graph.

the transition probabilities
can potentially change with temperature

State transition is
dependent only on the current state.

modeled as a *time inhomogeneous Markov chain*

Markov property in SA

- Transition Probability

$$P_{ij}(T) = \begin{cases} 0 & \text{if } j \notin N(i) \& j \neq i \\ \frac{1}{|N(i)|}\min(1, \exp\{(C(j) - C(i))/T\}) & \text{if } j \in N(i) \end{cases}$$

$$\text{for any } i \in V, P_{ij}(\text{at any temperature}) \geq \frac{1}{d}\exp\left(-\frac{\Delta}{T}\right) \text{ if } j \epsilon N(i)$$

$$P_{ii}(T) = 1 - \sum_{j \in N(i)} P_{ij}(T)$$

- And assume that G is strongly connected.

Fact 3.1.

Let X be the state of a Markov chain at time $t = t_0$.
The probability that a global minimum state is visited during the next q steps is dependent only on X and q and not on the states visited before.

Convergence of SA

- Notations
  - Let $N(i)$ be the set of neighbors of i.
  - T : the minmum temperature that SA was ever in
  - $\Delta = \max_{i \in V, j \in N(i)} \{C(i) - C(j)\}$
  - The degree of $G(V,E)$: d
  - The diameter of $G(V,E)$: D

Lemma 3.1.

If X is any state in V,
then the expected number of steps before a global optimal state is visited starting from X is $\leq \left(\frac{1}{d}\exp\left(-\frac{\Delta}{T}\right)\right)^{-D}$

Convergence of SA

proof.

Let g be any global optimal state.

There exists a directed path from X to g in $G(V, E)$ of length $q \leq D$.

Let $e_1, e_2, \ldots, e_q$ be the sequence of edges in the path.

Probability that g is visited starting from X is at least the probability that each one of the edges e_i, $1 \leq i \leq q$ is traversed in succession.

Therefore, this probability is at least $\left[\left(\frac{1}{d}\right)\exp\left(-\frac{\Delta}{T}\right)\right]^q \geq \left[\left(\frac{1}{d}\right)\exp\left(-\frac{\Delta}{T}\right)\right]^D$

Hence, The expected number of steps before g is visited is $\leq \left[d\ exp\left(\frac{\Delta}{T}\right)\right]^D$.

Convergence of SA

Theorem 3.1.

SA converges in time $\leq 2k\left[d\ exp\left(\frac{\Delta}{T}\right)\right]^D$, $with\ probability \geq \left(1-2^{-k}\right)$

,no matter what state is

proof.

Let $E = 2\left[d\ exp\left(\frac{\Delta}{T}\right)\right]^D$. We show that the probability of a global optimal state g not being visited kE steps is $\leq 2^{-k}$, by induction on k.

- **Induction Hypothesis.**
  - Irrespective of the start state,
    probability that g is not visited in kE steps is $\leq 2^{-k}$

Convergence of SA

proof.

- **Base case.**
  - When $k = 1$,
  - for any start state X, expected number of steps before g is visited is $\leq \frac{E}{2}$

    (by lemma 3.1)
- An application of Markov's inequality implies that the probability of g not being visited starting from X in E step is $\leq \frac{1}{2}$

If X is any non−negative random variable with mean p,
Markov's inequality implies that Prob.$[X > k\mu] \leq 1/k$, for any $k \rangle 0$.

Convergence of SA

proof.

- **Induction step.**
  - Assume the hypothesis for all $k \leq (r-1)$. We'll prove the hypothesis for $k = r$.

Let $X_E, X_{2E}, \ldots, X_{(r-1)E}$ be the states of the Markov chain during time steps $E, 2E, \ldots, (r-1)E$ respectively.

Let A be the event: g is not visited during the first E steps, and

B be the event: g is not visited during the next $(r-1)E$ steps.

Probability that g is not visited in rE steps, P, is given by

$$P = P[B|A] \times P[A]$$

Convergence of SA

proof.

Using fact 3.1,

$\mathrm{P}[B|A]$ depends on what state the Markov chain is in at time step E and the time duration $(r-1)E$.

$$P = P[A] \sum_{i \epsilon V} P[B|X_E = i] \times P[X_E = i]$$

since $P[A] \leq \frac{1}{2}$ and $P[B|X_E = i] \leq 2^{-(r-1)}$ for each $i \in V$,

$$P \leq \frac{1}{2} \cdot 2^{-(r-1)} = 2^{-r}$$

Convergence of SA

Corollary 3.1.

If Δ, T and d are assumed to be constants and $D = \theta(\log|V|)$ then SA converges in times polynomial in $|V|$ $with\ prob. \geq \left(1 - 2^{-\Omega|V|}\right)$

- Even if the system stays in the same temperature throughout, as long as enough time is given, the system will converge

Convergence of SA

Assume that the probability of generating state j from state i is $\frac{g(i)}{g(j)}$ where $g(i,j)$ is the 'weight' of j as a neighbor of i and $g(i)$ is a normalizing function such that $\sum_{j \in N(i)} g(i,j) = g(i)$

$$P_{ij}(T) = \begin{cases} 0 & \text{if } j \notin N(i) \& j \neq i \\ \frac{g(i,j)}{g(i)} \min(1, \exp\{(C(j) - C(i))/T\}) & \text{if } j \in N(i) \end{cases}$$

$$P_{ii}(T) = 1 - \sum_{j \in N(i)} P_{ij}(T).$$

Theorem 3.2.

SA converges in time $\leq 2k \left[\frac{1}{p} \exp\left(\frac{\Delta}{T}\right)\right]^D$ *with probability* $\geq (1 - 2^{-k})$, no matter what state is.

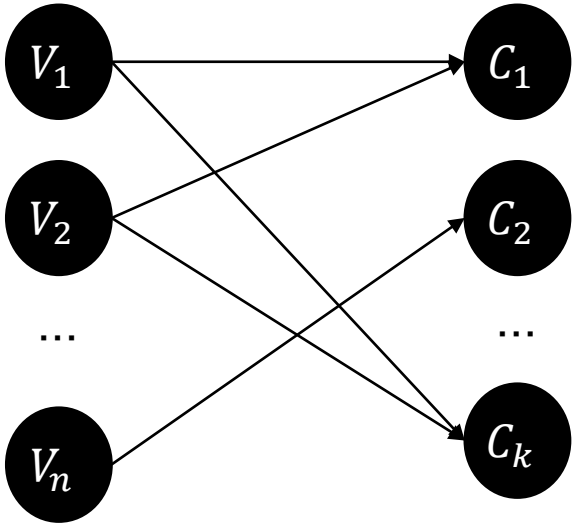
Cost functions with special properties

Definition 4.1.

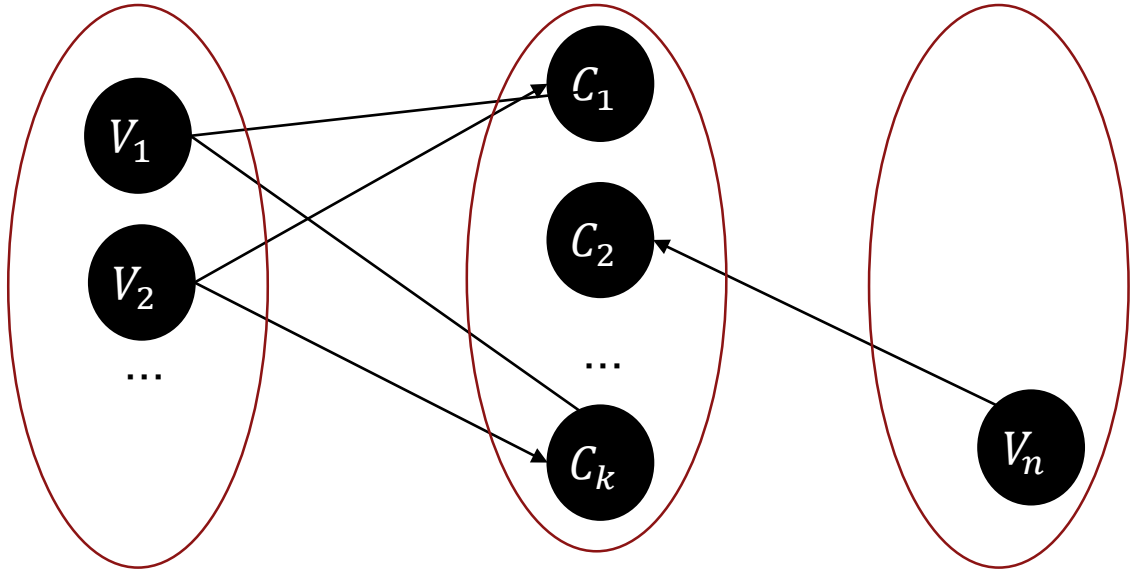
a graph $G(V,E)$ with n nodes is $'s(n) - separable'$ if there exist constants $a < 1$, $\beta > 0$ such that V can be partitioned into three subsets V_1, S, V_2. Also, no vertex in V_1 is adjacent to any vertex in V_2, where $|V_1|$, $|V_2| < \alpha n$, and $|S| < \beta s(n)$.
Moreover, the induced subgraphs of G on V_1, and on V_2 are s($|V_1|$)−separable and s($|V_2|$)−separable, respectively.

- By eliminating S from G, we can construction two roughly equal disjoint subgraphs.

Cost functions with special properties



Can be



Partition V_1 also $s|V_1| - separable$ Partition V_2 also $s|V_2| - separable$

Cost functions with special properties

Definition 4.2.

Suppose C is a cost function on n parameters $p_1, p_2, \ldots, p_n$. 'separability' of $C = C_1 + C_2 + \ldots + C_k$, where each C_i is a product of functions of the parameters. And we call each $C_i, 1 \leq i \leq$ as a clause.

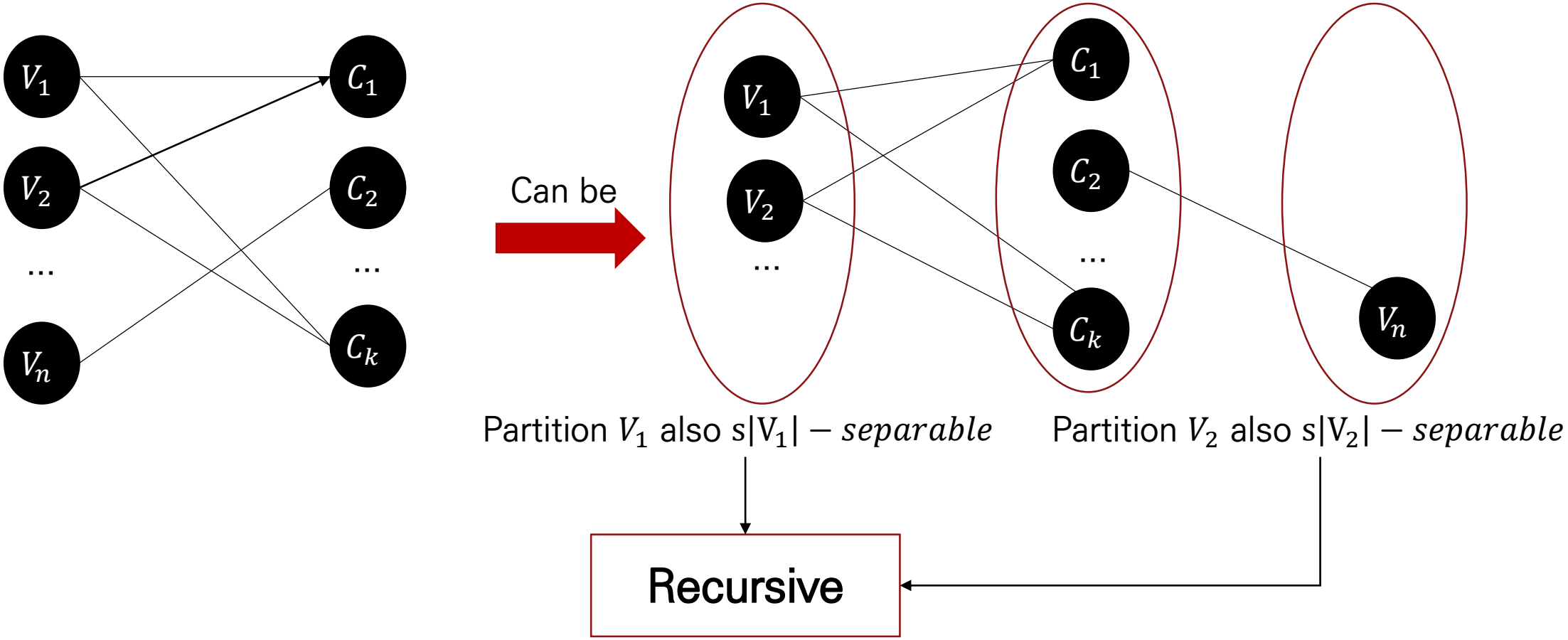
Definition 4.3.

Define a bipartite graph $G_c(V, E)$ whose nodes are the parameters and the clauses. There is an edge between a clause node and a parameter node if that parameter occurs in that clause. G_c is called the 'graph of C'. We say C is $s(n)$−separable if G_c is

Cost functions with special properties

small $s(n)$ of a cost function implies that
by assigning values to a small number of parameters we can obtain two independent subproblems such that the parameters involved in one subproblem are disjoint from the parameters of the other subproblem.

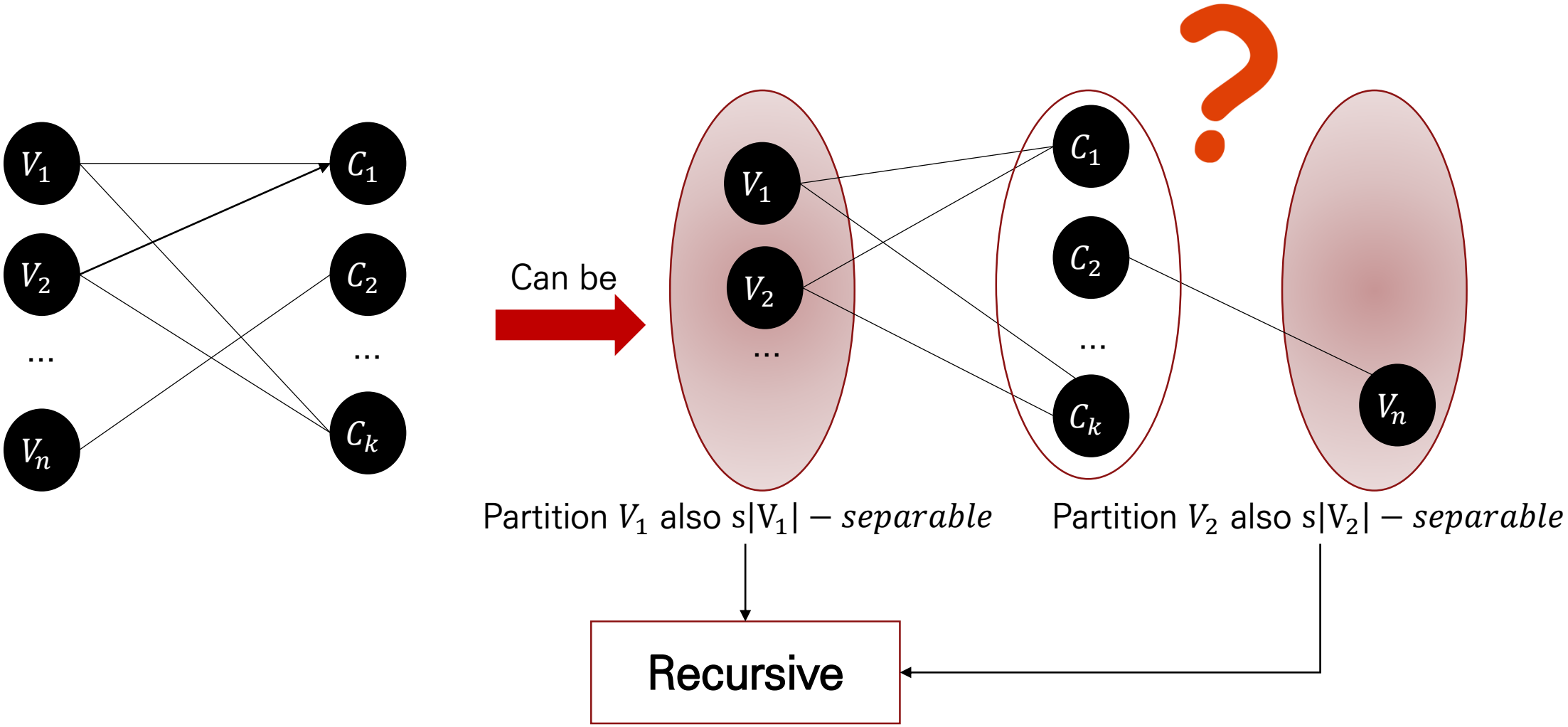
Cost functions with special properties



Design Nested-Annealing Algorithm

- One way of computing the minimum value of C
  - For each possible assignment of values to **parameters** in S, find the minimum value of C.
  - pick the minimum of these minima.
  - Finding the minimum of C under a particular assignment for S, is easy now.
- So, we need to find the minimum of two functions C_1 and C_2
  - where C_1 involves only parameters from V_1 and C_2 involve only parameters V_2
- Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be the restrictions of G on V_1 and V_2 respectively.
  - Finding the minimum of C_1 and C_2 can be done **recursively** by finding separators for G_1 and G_2 respectively.

Design Nested-Annealing Algorithm



Design Nested-Annealing Algorithm

- ***Suppose that an OP on $|S|$ parameters.***
  - There are thus $2^{|S|} < 2^{\beta s(n)}$ states of the OP.
  - The cost of each state : the minimum of C under that particular assignment to S.
  - Instead of considering each possible state of this OP, and computing the cost of each state, we can run a SA on this OP with $\leq \beta s(n)$ parameters.

Main Idea of Nested Annealing

Since SA algorithm only visits a small fraction of all possible states of the OP to come up with a quasi-optimal solution,
the number of states visited will be much less than $2^{\beta s(n)}$ in the OP with $|S|$ parameters.

Design Nested-Annealing Algorithm

procedure Nested_Annealing($G_C(V, E)$);

Find a separator set S for G_C. Let V_1, S, and V_2 be the partition of V. Also let G_1 and G_2 be restrictions of G on V_1 and V_2 respectively.
Find an optimal assignment for S by running an SA algorithm on these parameters. For each state of the corresponding Markov chain visited by SA we need to compute the cost.
To compute this cost, we need to find minimum of two other functions C_1 and C_2 (see the discussion above). Each of C_1 and C_2 involves $\leq \alpha n$ parameters.
Find these two minima recursively by finding separators for G_1 and G_2 respectively.

- $T(n)$: the expected run time of NA with n parameters.
  1. how many of the $2^{|S|}$ states at the top level will be visited
  2. for each state visited, the time needed to compute the cost of the state.

Time complexity of each step

1. the number of states visited will be no more than $2^{M(\beta(n))}$
2. accounting for a total expected cost of $\leq 2T(\alpha n)$.

$$T(n) \leq 2^{M(\beta S(n))} \cdot 2T(an)$$

$$\Rightarrow$$

$$T(n) \leq 2^{\sum_{i=0}^{\log n} M(\beta s(\alpha^i n))}$$

Design Nested-Annealing Algorithm

- If $s(n)$ is assumed to be $O(n^{\sigma})$ for some $\sigma < 1$
  - $T(n) \leq 2^{\gamma M(\beta s(n))} = 2^{O\left(M(\beta s(n))\right)}$ where $\gamma \leq \frac{1}{1-\sqrt{\alpha}}$
- And then Probability that the run time of Nested-Annealing exceeds kL is less than $\frac{1}{k}$, using Markov's inequality.

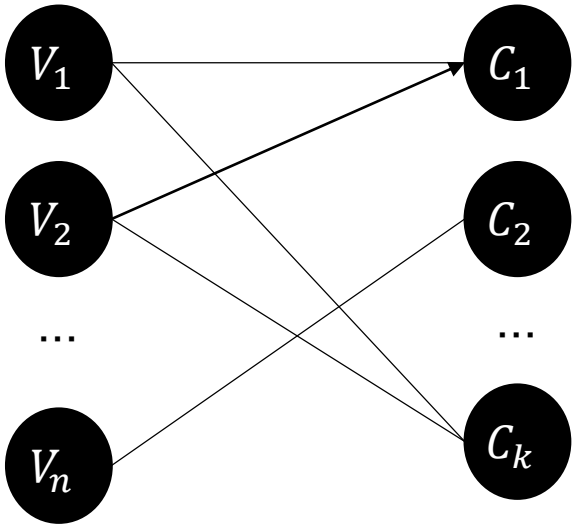
Theorem 4.1

NA converges in time $\leq n^{\alpha} L$ with probability $\geq (1 - n^{-\alpha})$.

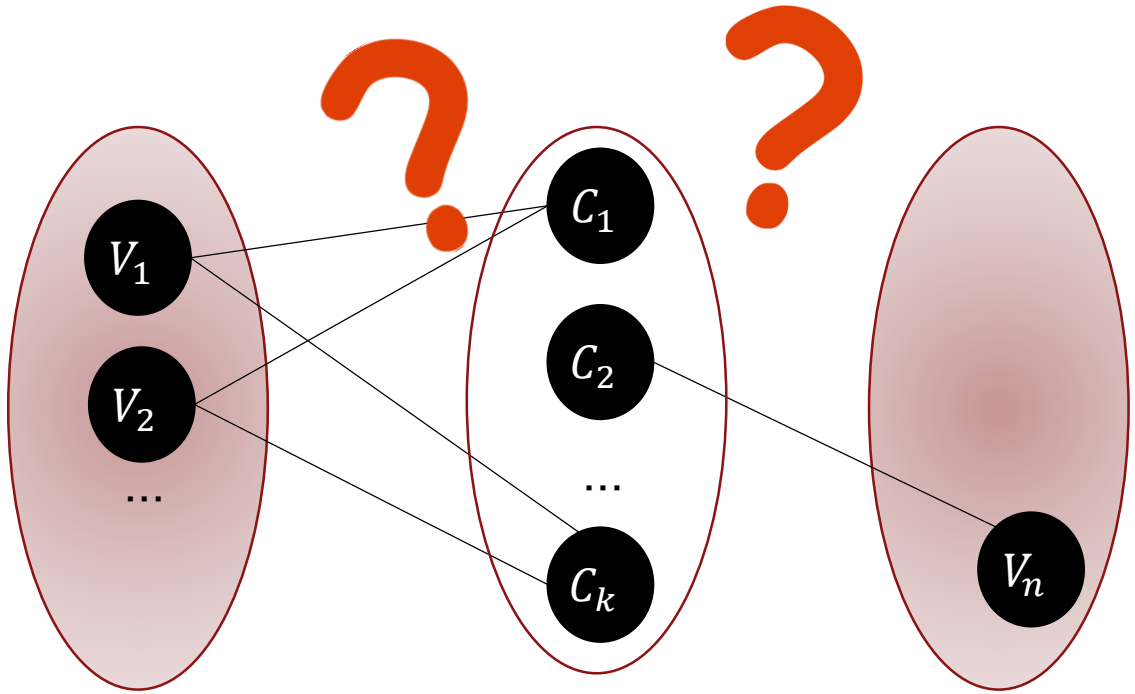
Corollary 4.1

If $M(n)$ is $O(n)$, NA converges in time $\leq n^{\alpha} 2^{O(s(n))}$ with probability $\geq (1 - n^{-\alpha})$.

Separability of Random Graphs



Can be



Partition V_1 also $s|V_1| - separable$

Partition V_2 also $s|V_2| - separable$

Recursive

Separability of Random Graphs

- Deciding if a given graph is $s(n)$−separable is NP−hard.
  - If $|S|$ is a constant fraction (or more) of $|V_1|$ and $|V_2|$, then there is no gain in running all the levels of recursion of the algorithm Nested Annealing.

Replace the instructions that call for computing the minimum of C_1 and C_2 recursively, with instructions to compute these two minima using SA

→ *Expected convergence time*
$= 2^{M(|S|)}[2^{M(|v_1|)} + 2^{M(|v_2|)}] = O(2^{M(|S|+\max[|V_1|,|V_2|])})$.

Separability of Random Graphs

Corollary 5.1

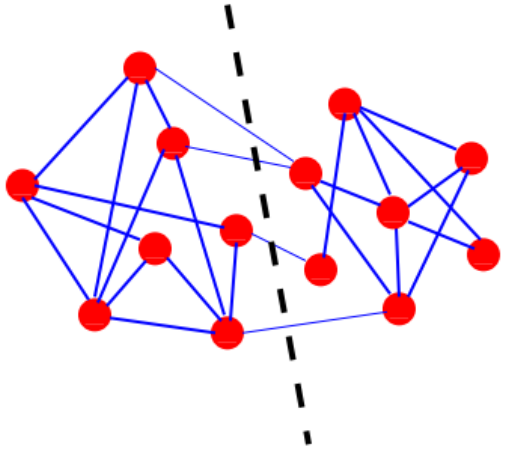
Almost every $G_p(V, E)$ is such that V can be partitioned into V_1, S, V_2 in such a way that there is no $V_1 - V_2$ edge and $|V_1| = \frac{1}{p}$. Also V_1 can be chosen to be any set of $\frac{1}{p}$ nodes.

- Using Corollary 5.1,
  Take any set of $\frac{1}{p}$ nodes as $V_1, \Gamma(V_1) - V_1$ as S, and $V - S - V_1$ as V_2.
  The separator set so found will be such that $|S| + \max[|V_1|, |V_2|]$ is no more than $n - \frac{1}{p}$.

Application of the algorithm for an NP−C/NP−hard example problem

- Graph partitioning problem
  - Given a graph G = (V, E) and
  - Find that partition $V = V_1 \cup V_2$ of V into equal sized sets that minimizes the number of edges that have end−points in different sets.

Figure 3: Graph Partitioning Problem



PROBLEM-SPECIFIC
1. What is a *solution*?
2. What are the *neighbors* of a solution?
3. What is the *cost* of a solution?
4. How do we determine an *initial* solution?

GENERIC
1. How do we determine an *initial temperature*?
2. How do we determine the *cooling ratio r*?
3. How do we determine the *temperature length L*?
4. How do we know when we are *frozen*?

Figure 5. Choices to be made in implementing simulated annealing.

Application of the algorithm for an NP−C/NP−hard example problem

- Problem Setting
  - **Solution**
    - any partition $V = V_1 \cup V_2$ of the vertex set (not just a partition into equal sized sets).

v_0	v_1							v_N
1	0	1	···	1	0	1	1	0

Figure 1: Solution representation of Graph coloring problem

  - Two partitions will be **neighbors**
    - if one can be obtained from the other by moving a single vertex from one of its sets to the other.

Figure 1: Neighbors of given state

1	0	1	···	1	0

⟷

1	0	1	···	1	1

Application of the algorithm for an NP–C/NP–hard example problem

- The **cost** of a partition (V_1, V_2)

$$c(V_1, V_2) = |\{u, v\} \in E\colon u \in V_1 \;\&\; v \in V_2| + \alpha(\,|V_1| - |V_2|)^2$$

where $|X|$ is the number of elements in set X
and α is a parameter called the imbalance factor.

- although this scheme allows infeasible partitions to be solutions, it penalizes them according to the square of the imbalance.

Consequently,
at low temperatures the solutions tend to be almost perfectly balanced

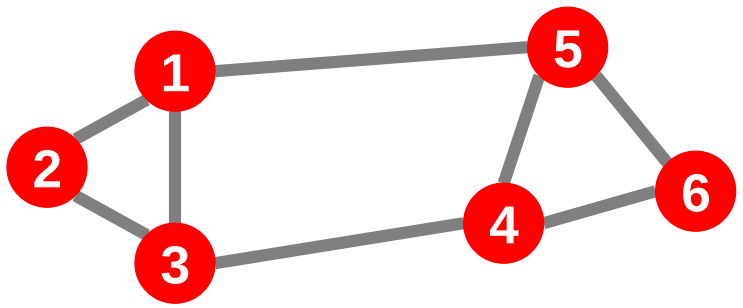
Application of the algorithm for an NP−C/NP−hard example problem

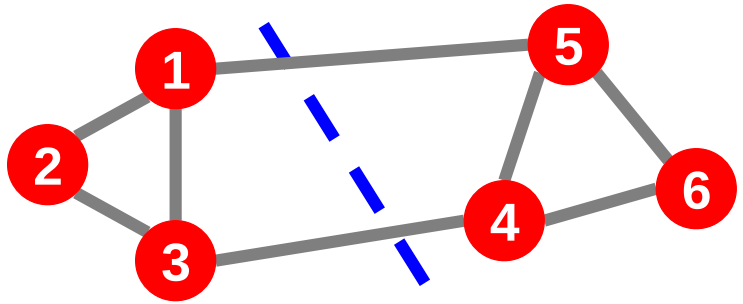
- Two final problem−specific details are our method
  - How to choose an initial solution
  - How to turn a non−feasible final solution into a feasible one.

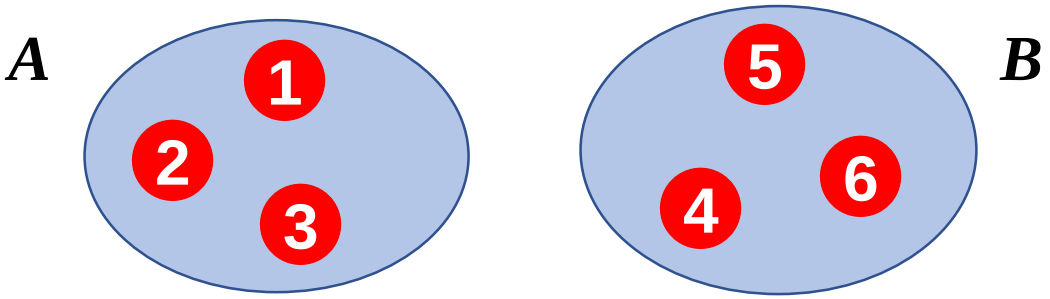
- Solution
  - First problem
    - Initial solutions are obtained by generating a random partition
      - e.g. for each vertex we flip an unbiased coin to determine whether it should go in V_1 or V_2
  - Second problem
    - Use a greedy heuristic to put it into balance if the final solution remains unbalanced
      - e.g. Find a vertex in the larger set that can be moved to the opposite set with the least increase in the cut−size, and move it.

Application of the algorithm for an NP–C/NP–hard example problem

1. Get an initial solution S.
2. Get an initial temperature $T > 0$.
3. While not yet *frozen* do the following.
   - 3.1 Perform the following loop L times.
     - 3.1.1 Pick a random neighbor S' of S.
     - 3.1.2 Let $\Delta = cost\ (S') - cost\ (S)$.
     - 3.1.3 If $\Delta \leqslant 0$ (downhill move), Set $S = S'$.
     - 3.1.4 If $\Delta > 0$ (uphill move), Set $S = S'$ with probability $e^{-\Delta/T}$.
   - 3.2 Set $T = rT$ (reduce temperature).
4. Return S.







- Is Frozen?
  - incremented by one
    - each time a temperature is completed for which the percentage of accepted moves is MINPERCENT or less
  - reset to 0
    - each time a solution is found that is better than the previous champion.

If the counter ever reaches 5,
we declare the process to be frozen.

References

[1] Rajasekaran, Sanguthevar. "On the convergence time of simulated annealing." (1990).

[2] "The Energy Distribution Function" *http://hyperphysics.phy-astr.gsu.edu/hbase/quantum/disfcn.html*

[3] Johnson, D.S, Aragon, C.R, McGeoch, L.A., and Schevon, C. “Optimization by Simulated annealing: an experimental evalution; PART 1, GRAPH PARTITIONING .” *Operations Resarch*, 39(3):378–406, 1991.

[4] Du, Ke-Lin, and M. N. S. Swamy. "Simulated Annealing." *Search and Optimization by Metaheuristics*. Springer International Publishing, 2016. 29–36.

[5] Garey, Michael R., and David S. Johnson. *Computers and intractability*. Vol. 29. New York: wh freeman, 2002.