Measures of Fault Tolerance in Distributed Simulated Annealing

Aaditya Prakash

Infosys Limited
prakash@aaditya.info

International Conference on Perspective of Computer Confluence with Sciences, 2012

Outline

Outline

Probabilistic and Meta-heuristic Algorithm.

Similar to Annealing in Metallurgy

- $P(E) = e^{\frac{-E}{kT}}$. where, $P(E)$ is Energy Function, T is Temperature, k is Boltzmann constant
- Energy Function has high value at Higher Temperature.
- Uses Metropolis-Hastings algorithm to generate its sample space.

Probabilistic and Meta-heuristic Algorithm.

Similar to Annealing in Metallurgy

- $P(E) = e^{\frac{-E}{kT}}$. where, $P(E)$ is Energy Function, T is Temperature, k is Boltzmann constant
- Energy Function has high value at Higher Temperature.
- Uses Metropolis-Hastings algorithm to generate its sample space.

Probabilistic and Meta-heuristic Algorithm.

Similar to Annealing in Metallurgy

- $P(E) = e^{\frac{-E}{kT}}$. where, $P(E)$ is Energy Function, T is Temperature, k is Boltzmann constant
- Energy Function has high value at Higher Temperature.
- Uses Metropolis-Hastings algorithm to generate its sample space.

Outline

Algorithm

- Start with the system in a known configuration, at known energy E.
- while T is High {
- Perturb system slightly (goto new location on search space)
- Compute E, change in energy due to perturbation
- if($\Delta E < 0$) then accept this perturbation, this is the new system
- else accept this system with probability equal to Energy Function $P(E)$
- }
- stop when equilibrium is reached or T is low

Algorithm

- Start with the system in a known configuration, at known energy E.
- while T is High {
- Perturb system slightly (goto new location on search space)
- Compute E, change in energy due to perturbation
- if($\Delta E < 0$) then accept this perturbation, this is the new system
- else accept this system with probability equal to Energy Function $P(E)$
- }
- stop when equilibrium is reached or T is low

Algorithm

- Start with the system in a known configuration, at known energy E.
- while T is High {
- Perturb system slightly (goto new location on search space)
- Compute E, change in energy due to perturbation
- if($\Delta E < 0$) then accept this perturbation, this is the new system
- else accept this system with probability equal to Energy Function $P(E)$
- }
- stop when equilibrium is reached or T is low

Algorithm

- Start with the system in a known configuration, at known energy E.
- while T is High {
- Perturb system slightly (goto new location on search space)
- Compute E, change in energy due to perturbation
- if($\Delta E < 0$) then accept this perturbation, this is the new system
- else accept this system with probability equal to Energy Function $P(E)$
- }
- stop when equilibrium is reached or T is low

Algorithm

- Start with the system in a known configuration, at known energy E.
- while T is High {
- Perturb system slightly (goto new location on search space)
- Compute E, change in energy due to perturbation
- if($\Delta E < 0$) then accept this perturbation, this is the new system
- else accept this system with probability equal to Energy Function $P(E)$
- }
- stop when equilibrium is reached or T is low

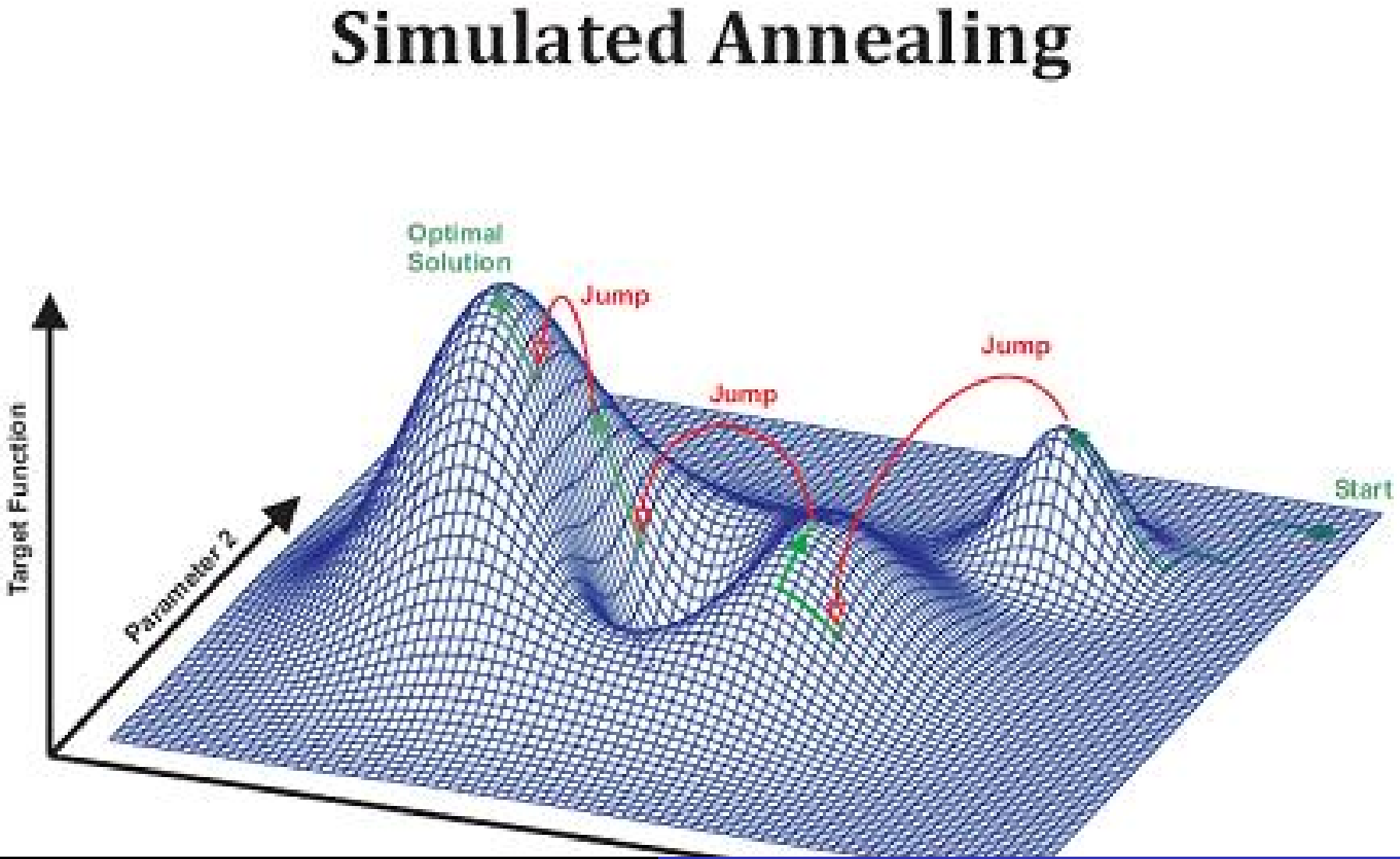
Algorithm

- Start with the system in a known configuration, at known energy E.
- while T is High {
- Perturb system slightly (goto new location on search space)
- Compute E, change in energy due to perturbation
- if($\Delta E < 0$) then accept this perturbation, this is the new system
- else accept this system with probability equal to Energy Function $P(E)$
- }
- stop when equilibrium is reached or T is low

Algorithm

- Start with the system in a known configuration, at known energy E.
- while T is High {
- Perturb system slightly (goto new location on search space)
- Compute E, change in energy due to perturbation
- if($\Delta E < 0$) then accept this perturbation, this is the new system
- else accept this system with probability equal to Energy Function $P(E)$
- }
- stop when equilibrium is reached or T is low

Search Space
Problem of local optima



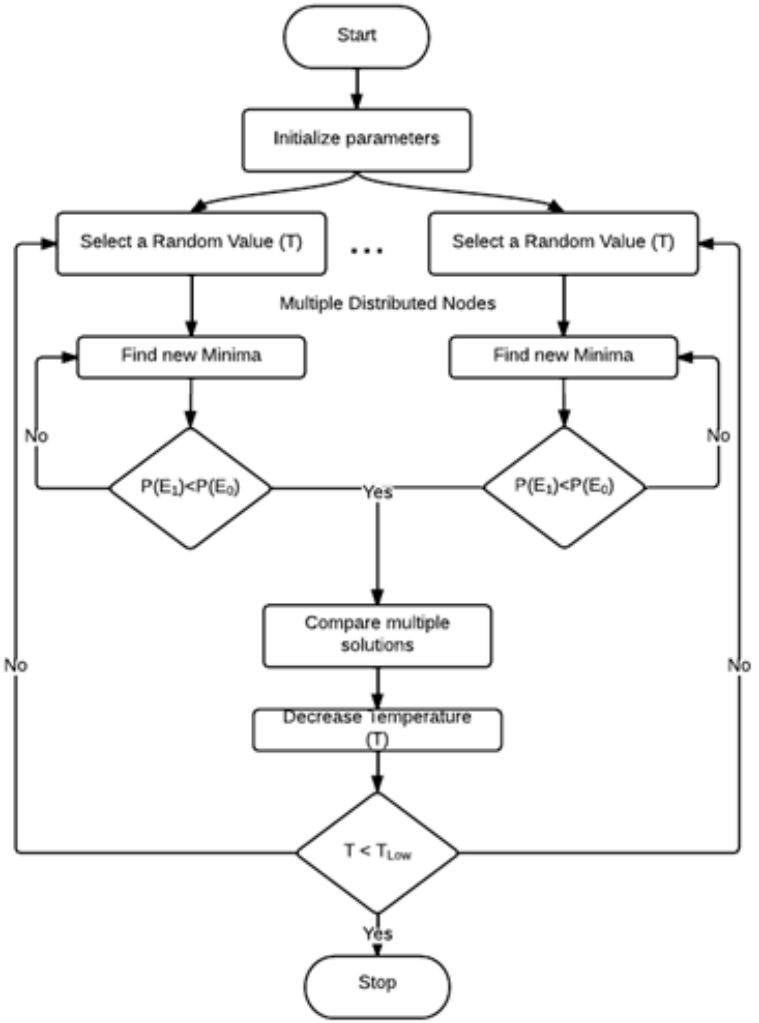
Outline

Distributed Simulated Annealing (DSA)

- MapReduce
  - (Radesnki 2012)
- CUDA
  - (Zbierski 2011)
- OpenCL
  - (Choong 2010)

DSA Algorithm

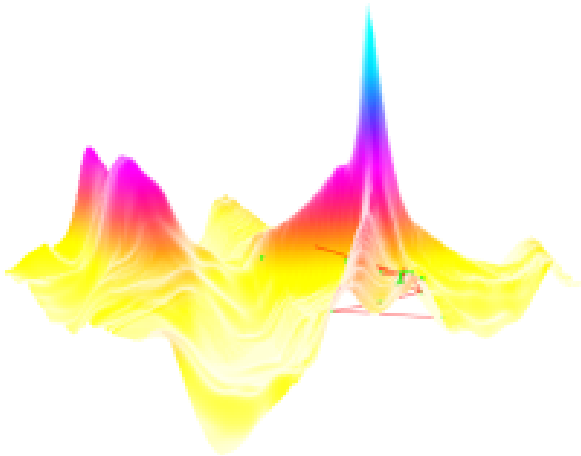
- Master/Host - Compare
- Cluster/Device - Search Solution



Outline

Sources of Design Faults

- Design Faults
  - Difficult Search Space
  - No memory of best solution (unlike Tabu search)
  - Pseudo Random Number Generator



Outline

Sources of Operational Faults

Type of failure	Description
Crash failure	A server halts, but is working correctly until it halts
Omission failure *Receive omission* *Send omission*	A server fails to respond to incoming requests A server fails to receive incoming messages A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure *Value failure* *State transition failure*	The server's response is incorrect The value of the response is wrong The server deviates from the correct flow of control
Arbitrary failure (Byzantine failure)	A server may produce arbitrary responses at arbitrary times

Source: Lecture Notes- Prof. Jalal Y. Kawash at Univ. of Calgary

---

Independent Failure
-Loss of Node and Loss of Data
-solved by design of MapReduce

Sources of Operational Faults

Type of failure	Description
Crash failure	A server halts, but is working correctly until it halts
Omission failure *Receive omission* *Send omission*	A server fails to respond to incoming requests A server fails to receive incoming messages A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure *Value failure* *State transition failure*	The server's response is incorrect The value of the response is wrong The server deviates from the correct flow of control
Arbitrary failure (Byzantine failure)	A server may produce arbitrary responses at arbitrary times

Source: Lecture Notes- Prof. Jalal Y. Kawash at Univ. of Calgary

---

Independent Failure

-Loss of Node and Loss of Data

-solved by design of MapReduce

Outline

Communication Faults

- Unreliable Communication
  - Incorrect result
- Insecure Communication
  - Incorrect result
- Costly Communication
  - Poor Performance
  - If the overhead of communication of nodes exceed the ratio of fraction of work to total Speedup then benefits of distribution of optimization is highly compromised
    - Amdahl's Law $\frac{1}{(1-P)+\frac{P}{S}}$

Communication Faults

- Unreliable Communication
  - Incorrect result
- Insecure Communication
  - Incorrect result
- Costly Communication
  - Poor Performance
  - If the overhead of communication of nodes exceed the ratio of fraction of work to total Speedup then benefits of distribution of optimization is highly compromised
    - Amdahl's Law $\frac{1}{(1-P)+\frac{P}{S}}$

Communication Faults

- Unreliable Communication
  - Incorrect result
- Insecure Communication
  - Incorrect result
- Costly Communication
  - Poor Performance
  - If the overhead of communication of nodes exceed the ratio of fraction of work to total Speedup then benefits of distribution of optimization is highly compromised
    - Amdahl's Law $\frac{1}{(1-P)+\frac{P}{S}}$

Outline

Tolerance

- Adaptive vs Strategic
  - Flexible Adaptive Tolerant System
    - Can handle unprecedented failures
  - Strategic Fault Tolerance
    - Predictive handling
    - (Marin et al 2001 - Flexible)
  - Pooling of Search Space - Futile
    - Stochastic Search
  - Hashing of Intermediate results
    - No guarantee of having searched but quick ($O(n)$) verification
    - MapReduce - fast at hashing
    - Ganjisaffar et al, Tunning of MapReduce for DSA, achieved AUC $> 90\%$

Tolerance

- Adaptive vs Strategic
  - Flexible Adaptive Tolerant System
    - Can handle unprecedented failures
  - Strategic Fault Tolerance
    - Predictive handling
    - (Marin et al 2001 - Flexible)
  - Pooling of Search Space - Futile
    - Stochastic Search
  - Hashing of Intermediate results
    - No guarantee of having searched but quick ($O(n)$) verification
    - MapReduce - fast at hashing
    - Ganjisaffar et al, Tunning of MapReduce for DSA, achieved AUC $> 90\%$

Tolerance

- Adaptive vs Strategic
  - Flexible Adaptive Tolerant System
    - Can handle unprecedented failures
  - Strategic Fault Tolerance
    - Predictive handling
    - (Marin et al 2001 - Flexible)
  - Pooling of Search Space - Futile
    - Stochastic Search
  - Hashing of Intermediate results
    - No guarantee of having searched but quick ($O(n)$) verification
    - MapReduce - fast at hashing
    - Ganjisaffar et al, Tunning of MapReduce for DSA, achieved AUC $> 90\%$

Tolerance

- Adaptive vs Strategic
  - Flexible Adaptive Tolerant System
    - Can handle unprecedented failures
  - Strategic Fault Tolerance
    - Predictive handling
    - (Marin et al 2001 - Flexible)
  - Pooling of Search Space - Futile
    - Stochastic Search
  - Hashing of Intermediate results
    - No guarantee of having searched but quick ($O(n)$) verification
    - MapReduce - fast at hashing
    - Ganjisaffar et al, Tunning of MapReduce for DSA, achieved AUC $> 90\%$

Outline

Recovery

- Cluster Replacement
  - Cold/Warm Standby - No use
    - Cannot perform backward error recovery
    - If temperature is still High, next search sequence is as good as any other
  - **Hybrid Replication Mechanism**
    - If Temperature is High - No result replication or broadcast
    - -Saves lot of time and space
    - If Temperature is Low ($T < T_{Low}$), convert some searching Node to reciprocating Nodes
    - Ensures when solution is found and if Node is dead, we will have a copy of the solution
    - -Reasoning: Higher Probability of finding optimal solution at lower T. Remember $P(E)$.
- Anomaly Node Detection
  - Several Machine Learning algorithms to detect anomalous

Recovery

- Cluster Replacement
  - Cold/Warm Standby - No use
    - Cannot perform backward error recovery
    - If temperature is still High, next search sequence is as good as any other
  - **Hybrid Replication Mechanism**
    - If Temperature is High - No result replication or broadcast
    - -Saves lot of time and space
    - If Temperature is Low ($T < T_{Low}$), convert some searching Node to reciprocating Nodes
    - Ensures when solution is found and if Node is dead, we will have a copy of the solution
    - -Reasoning: Higher Probability of finding optimal solution at lower T. Remember $P(E)$.
- Anomaly Node Detection
  - Several Machine Learning algorithms to detect anomalous

References

1. S. Kierkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. Science, 220: 671–680, 1983
2. Muhammad Arshad and Marius C. Silaghi. Distributed Simulated Annealing. In Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems, volume 112 of Frontiers in Artificial Intelligence and Applications. IOS Press, 2004.
3. Krishan, K. Ganeshan, and Ram, D. Janaki. Distributed simulated annealing algorithms for job shop scheduling. IEEE Trans. Systems Man Cybernet. 25, 7 (July 1995), 1102–1109
4. Atanas Radenski. 2012. Distributed simulated annealing with mapreduce. In Proceedings of the 2012t European conference on Applications of Evolutionary Computation(EvoApplications'12). Springer-Verlag.
5. F. Glover and C. McMillan (1986). "The general employee scheduling problem: an integration of MS and AI". Computers and Operations Research.
6. YANG, C., YEN, C., TAN, C., AND MADDEN, S. Osprey: (2010) Implementing MapReduce-style fault tolerance in a shared-nothing distributed database, ICDE.
7. Capiluppi M. (2007). Fault Tolerance in Large Scale Systems: Hybrid and distributed Approaches. Ph.D. Thesis, University of Bologna, Italy.
8. Rodgers, David P. (June 1985). "Improvements in multiprocessor system design". ACM SIGARCH Computer Architecture News archive (New York, NY, USA: ACM) 13 (3): 225–231.
9. Ganeshan, K. Designing and implementing flexible distributed problem solving systems. M.S. Thesis, Department of Computer Science and Engineering, Indian Institute of Technology, Madras, 1993
10. Ganjisaffar Y., Debeauvais T., Javanmardi S., Caruana R., Lopes C., Distributed tuning of machine learning algorithms using MapReduce clusters, Proceedings of the KDD 2011 Workshop on Large-scale Data Mining, San Diego, 2011, pages 1-8.