Simulated Annealing for Placement Problem to minimise the wire length

Ajay Sharma

2nd May 05
Chapter 1

Algorithm

1.1 Definition of Algorithm

A step by step procedure to solve any problem is called as Algorithm

1.1.1 Algorithm

An example of such an algorithm is Adding members of an array. The algorithm for such an example would be:

```
start
integer i;
integer result;
array S[n];

result = 0;
for (i=1;i<=n;i++)
    result=result+ S[i];
return result;
end
```

We will talk about Simulated Annealing in detail and use it for solving VLSI CAD problems.
Chapter 2

Simulated Annealing

Before we understand the Algorithm lets try to understand the Randomized algorithm.

2.1 Randomized Algorithm

A Randomized Algorithm is an algorithm uses random bits as an auxilliary input to guide its behaviour, in the hope of achieving a good performance in the "average case".

An Example of this would be finding 'a' in an array of 'n' elements given that half of them are 'a's and half are 'b's. The obvious approach is to look for all elements of array but that will take 'n/2' operations. But at random we can find 'a' at a much higher probability than other method.

Lets now begin with Simulated annealing.

2.2 Simulated Annealing

In Short, Simulated Annealing(SA) is a generic probabilistic meta-algorithm for the global optimization problem namely locating a good approximation to the global optimum of a given function in a large search space. By global optimum i mean selection from a domain that yeilds a highest value when a specific function is given. For example the function : f(x)=-x^2 + 2, is defined over the real number axis and has the highest value when x=0, ie: f(x)=2.
The search space is the set of all possible solutions to the problems.


The inspiration comes from metallurgy where we do a lot of annealing. In annealing we heat a solid and do controlled cooling to increase the size of the crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions and wander randomly through states of higher energy; slow cooling gives them chance to come to a state of lower energy than the initial one.
2.2.1 Basic Iteration

We decide between moving the system to the neighbouring state $s'$ from the state $s$ based on the probability that the system ultimately reaches the lowest possible energy state.

We do it till we get the best possible solution or when the time bound is over or resources exhausted.

2.2.2 Neighbours of the state

The neighbours of the state are specified by the user.

2.2.3 Transition Probability

Probability of making a transition is decided based on the function $P(\Delta E, T)$ where $\Delta E = E(S') - E(S)$ ..... $T$ :- Temperature.

Sometimes we swap the system state from $S$ to $S'$ even if the next state is not having lower energy level which means we don't stay at a false minimum. Where the neighbours are at high energy level but we still swap in the hope that eventually we may reach a lower energy state. We see that as:

For small values of $T$ the system prefers to go downhill than uphill.

When Time=0, the system is greedy algorithm, i.e.: moves to the next state iff the next state is of lower energy level. At higher Temperature coarse energy variation cause system change and at lower temperature finer energy variation cause system change.

2.2.4 Annealing schedule

Initially the system assumes the highest possible energy level and then it is gradually decreased.

2.2.5 PSEUDO CODE

1. $S=S_0$ – Initial state 'S' = $S_0$ (state 0)
2. \( E = E(S) \) – Energy is equal to energy of the state

3. \( k = 0 \) – step = 0 ie. initial step.

4. while \( k < K_{\text{max}} \) & \( E : e_{\text{max}} \) – Repeat the loop till max steps and lowest possible energy.

5. \( s_n := \text{neighbours}(S) \) – this should generate randomly generated neighbour. \( S_n \) :- neighbour’s state.

6. \( E(n) = E(S_n) \) – Energy is the neighbours energy. \( E(sn) \) is the energy of the neighbour state..

7. if random() (lessthan) \( P(en - e, \text{temp}(K/K_{\text{max}})) \) then – means random calls generates random values between \([0,1]\) so check for change in energy over the over the change in temperature.

8. \( s = s_n ; e = e_n \) – state is the next state, energy is the next state.

9. \( k = k + 1 \) – step increase.

10. return \( S \) – return the current state as output of the algorithm

### 2.2.6 Parameters

1. 1. State Space: The State machine or the perhaps the state.

2. 2. Neighbour selection method: Enumerating the next state candidate.

3. 3. Probability Transition Function: The probability of Transition.

4. 4. Annealing Schedule: Temperature decrease schedule.

   Its more of an Art than Science.

### 2.2.7 State Neighbours

Choosing the neighbours is very critical, always swap adjacent neighbours state not any arbitrary state.

Chances are more likely that energy increases when we swap any arbitrary state, rather than adjacent swap.

More than that at Higher temperature coarser moves allowed and at lower temperature narrower moves allowed.
2.2.8 Transition Probability

It is not as critical as the neighbourhood graph provided follows the requirement of SA.

2.2.9 Classical Formula

If $\Delta E = -\text{ve}$ (energy is reduced) then the probability of $P(\Delta E, T)$ is accepted is '1' the move is accepted. Otherwise the Probability is $= e^{\frac{-\Delta E}{T}}$. from metropolis-Hastings theorem used to generate samples from Maxwell's Boltzman distribution.

2.2.10 Annealing Schedule

The initial temperature should be large enough so that probability of accepting uphill/downhill moves are same. At any random state, $\Delta E$ should be estimated.

The temperature should be finally decreased to 0 or nearly zero. Popular choice is the Exponential schedule ie. temperature decrease by a fixed factor. $\alpha^{t+1}$.
Chapter 3

C++ Based Implementation

The following is the program that i am working on . It has the following :

1. Input
   (a) BDNET file
   (b) No of iterations at each temperature
   (c) No of Temperature points to try

2. Output
   (a) Placement of the optimized Cells
   (b) Plot of the Cost function
3.1 BDNET File

Netlist files are listed in this form. This is the sample BDNET ckt and the BDNET file is given below.
MODEL sample:unplaced;
TECHNOLOGY scmos;
VIEWTYPE SYMBOLIC;
EDITWTYLE SYMBOLIC;

INPUT clk,a1;
OUTPUT z<1:0>;
SUPPLY Vdd;
GROUND GND;
INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/nanf251": physical NAME: "u1" "GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "n0";
"A1" : "a1";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/anf251": physical NAME: "u2" "GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "z0";
"A1" : "clk";
"B!" : "n0";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/anf251": physical NAME: "u3"
"GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "z1";
"A1" : "n0";
"B!" : "clk";
Chapter 4

Circuit: Sample1

We start with testing the circuits and finding the cost reduction and CPU time required. The first circuit is the smallest and the last is the biggest. These Tests are performed on the Test Computer with Linux OS/intel 1.7GHZ Celeron/128 MB RAM/845 Chipset Motherboard.

4.1 Specification

1. Total Size : 14 Cells.
2. Total Nets : 17
3. Initial Cost : 29
4. Final Cost : 23, 22, 23
5. Initial temperature : 10000
6. Percent Reduction in Cost = 26 percent
7. Iteration = 150
8. Temperature Steps : 100
4.2 Initial Placement

This is the initial plot and the wire length is marked on the nets. Total Length is 29 initially.
4.3 Final Placement

This is the Final plot and the wire length is reduced from 29 to 23.
4.4 Cost Function Plot

4.5 VERILOG CODE

module sample2(A,B,C,F,G);
input A,B,C;
output F,G;
wire n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12;

not u1(n1,A);
not u2(n2,B);

or u3(n3,n1,n2);
and u4(n4,A,n3);
and u5(n5,n3,B);

or u6(n6,n4,n5);
not u7(n7,n6);
not u8(n8,C);

or u9(n9,n7,n8);
and u10(n10,n6,n9);
and u11(n11,n9,C);
and u12(n12,n9,n3);

or u13(F,n10,n11);
not u14(G,n12);

endmodule

4.6 BDNET NETLIST

MODEL "sample":unplaced;
TECHNOLOGY scmos;
VIEWTYPE SYMBOLIC;
EDITSTYLE SYMBOLIC;
OUTPUT "F" : "F";
OUTPUT "G" : "G";

INPUT "A" : "A";
INPUT "B" : "B";
INPUT "C" : "C";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/invf101":physical NAME = "U1" 
"GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "n1";
"A1" : "A";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/invf101":physical NAME = "U2" 
"GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "n2";
"A1" : "B";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/norf201":physical NAME = "U3" 
"GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "n3";
"A1" : "n1";
"B1" : "n2";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/blf00001":physical NAME = "U4" 
"GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "n4";
"A1" : "A";
"B1" : "n3";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/blf00001":physical NAME = "U5" 
"GND!" : UNCONNECTED;
INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/blf00001":physical NAME = "U11"
"GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "n11";
"A1" : "n9";
"B1" : "C";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/blf00001":physical NAME = "U12"
"GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "n12";
"A1" : "n9";
"B1" : "n3";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/norf201":physical NAME = "U13"
"GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "nF";
"A1" : "n10";
"B1" : "n11";

INSTANCE "$OCTTOOLS/tech/scmos/msu/stdcell2_2/invf101":physical NAME = "U14"
"GND!" : UNCONNECTED;
"Vdd!" : UNCONNECTED;
"O" : "G";
"A1" : "n12";

ENDMODEL;
Chapter 5

Sample Circuit 2: 8 bit Full Adder with Register

5.1 Specification

1. Total Size : 73 Cells.
2. Total Nets : 91
3. Initial Cost : 505
4. Final Cost : 154,158,153
5. Initial temperature: 10000
6. Percent Reduction in Cost = 226 percent
7. Iteration = 150
8. Temperature Steps : 100
5.2 Final Placement

<table>
<thead>
<tr>
<th></th>
<th>reg1</th>
<th>reg2</th>
<th>u32</th>
<th>u51</th>
<th>u25</th>
<th>u15</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>u34</td>
<td>u14</td>
<td>u35</td>
<td>u10</td>
<td>u52</td>
<td>u48</td>
<td>x</td>
</tr>
<tr>
<td>u20</td>
<td>u27</td>
<td>u36</td>
<td>u50</td>
<td>u53</td>
<td>u7</td>
<td>u30</td>
<td>reg3</td>
</tr>
<tr>
<td>x</td>
<td>u5</td>
<td>u49</td>
<td>u41</td>
<td>u40</td>
<td>u55</td>
<td>u23</td>
<td>u54</td>
</tr>
<tr>
<td>u19</td>
<td>u57</td>
<td>u38</td>
<td>u13</td>
<td>u39</td>
<td>u56</td>
<td>u45</td>
<td>u16</td>
</tr>
<tr>
<td>u26</td>
<td>u58</td>
<td>u37</td>
<td>u21</td>
<td>u28</td>
<td>reg0</td>
<td>u46</td>
<td>x</td>
</tr>
<tr>
<td>u33</td>
<td>reg6</td>
<td>reg7</td>
<td>u4</td>
<td>reg5</td>
<td>u62</td>
<td>reg4</td>
<td>u59</td>
</tr>
<tr>
<td>u8</td>
<td>u12</td>
<td>u24</td>
<td>u31</td>
<td>u63</td>
<td>u44</td>
<td>u29</td>
<td>u66</td>
</tr>
<tr>
<td>u67</td>
<td>u68</td>
<td>u18</td>
<td>u65</td>
<td>u9</td>
<td>u17</td>
<td>u22</td>
<td>u65</td>
</tr>
<tr>
<td>u11</td>
<td>u61</td>
<td>x</td>
<td>u60</td>
<td>u64</td>
<td>u43</td>
<td>u42</td>
<td>x</td>
</tr>
</tbody>
</table>

5.3 Cost Function Plot

5.4 VERILOG CODE

The verilog Code can be found at the website: www.csun.edu/ags55111. Its too bog to be included in the project report.
Chapter 6

Sample Circuit 3

6.1 Specification

1. Total Size : 93 Cells.
2. Total Nets : 114
3. Initial Cost : 405
4. Final Cost : 138,144,140
5. Initial temperature: 10000
6. Percent Reduction in Cost = 189 percent
7. Iteration = 150
8. Temperature Steps : 100
6.2 Final Placement

<table>
<thead>
<tr>
<th>u8</th>
<th>u6</th>
<th>u15</th>
<th>u5</th>
<th>u11</th>
<th>u18</th>
<th>u12</th>
<th>u2</th>
<th>u13</th>
</tr>
</thead>
<tbody>
<tr>
<td>u16</td>
<td>u7</td>
<td>u21</td>
<td>u4</td>
<td>x</td>
<td>u24</td>
<td>u38</td>
<td>u40</td>
<td>u1</td>
</tr>
<tr>
<td>u22</td>
<td>u20</td>
<td>u25</td>
<td>u14</td>
<td>u3</td>
<td>u37</td>
<td>u47</td>
<td>u48</td>
<td>u39</td>
</tr>
<tr>
<td>x</td>
<td>u26</td>
<td>u28</td>
<td>u56</td>
<td>u27</td>
<td>u60</td>
<td>u71</td>
<td>u61</td>
<td>u39</td>
</tr>
<tr>
<td>u19</td>
<td>u33</td>
<td>x</td>
<td>u57</td>
<td>u23</td>
<td>u82</td>
<td>u76</td>
<td>u70</td>
<td>u58</td>
</tr>
<tr>
<td>u34</td>
<td>u45</td>
<td>u55</td>
<td>u69</td>
<td>u17</td>
<td>u88</td>
<td>u83</td>
<td>u62</td>
<td>u59</td>
</tr>
<tr>
<td>u31</td>
<td>u54</td>
<td>u68</td>
<td>u75</td>
<td>u9</td>
<td>u91</td>
<td>x</td>
<td>u63</td>
<td>u36</td>
</tr>
<tr>
<td>u44</td>
<td>u32</td>
<td>u80</td>
<td>u81</td>
<td>u10</td>
<td>u89</td>
<td>u84</td>
<td>u72</td>
<td>u46</td>
</tr>
<tr>
<td>u30</td>
<td>u29</td>
<td>u43</td>
<td>u87</td>
<td>u90</td>
<td>u92</td>
<td>u85</td>
<td>u77</td>
<td>u41</td>
</tr>
<tr>
<td>u52</td>
<td>u51</td>
<td>u50</td>
<td>u78</td>
<td>u86</td>
<td>x</td>
<td>u42</td>
<td>u49</td>
<td>u41</td>
</tr>
<tr>
<td>u53</td>
<td>u67</td>
<td>u66</td>
<td>u74</td>
<td>u79</td>
<td>x</td>
<td>u65</td>
<td>u73</td>
<td>u64</td>
</tr>
</tbody>
</table>

6.3 Cost Function Plot

6.4 VERILOG CODE

The verilog Code can be found at the website: www.csun.edu/ags55111. Its too bog to be included in the project report.
Chapter 7

Sample Circuit 4: ALU

7.1 Specification

2. Total Nets : 400
3. Initial Cost : 7499
4. Final Cost : 1633,1624,1688
5. Initial temperature: 10000
6. Percent Reduction in Cost = 355 percent
7. Iteration = 150
8. Temperature Steps : 100

7.2 Final Placement

The final placement file is the alu.dat and can be found at www.csun.edu/ags55111. It again is too big to be included in the project report.
7.3 Cost Function Plot

7.4 VERILOG CODE

The verilog Code can be found at the website: www.csun.edu/ags55111. Its too big to be included in the project report.

7.5 BDNET File

Can be found at www.csun.edu/ags55111
Chapter 8

Sample Circuit 5: 8x8 Bit Multiplier

8.1 Specification

1. Total Size : 441 Cells.
2. Total Nets : 457
3. Initial Cost : 7105
4. Final Cost : 1366,1409,1324
5. Initial temperature: 10000
6. Percent Reduction in Cost = 420 percent
7. Iteration = 150
8. Temperature Steps : 100

8.2 Final Placement

The final placement file is the alu.dat and can be found at www.csun.edu/ags55111
It again is too big to be included in the project report.
8.3 Cost Function Plot

8.4 VERILOG CODE

The verilog Code can be found at the website : www.csun.edu/ ags55111. Its too big to be included in the project report.

8.5 BDNET File

Can be found at www.csun.edu/ ags55111
Chapter 9

Final Analysis

I have used the Pentium Celeron 1.7GHz/Linux/128 RAM/845 chipset computer for all the test simulation. The summary of results is as follows:

<table>
<thead>
<tr>
<th>Ckt</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cells</td>
<td>14</td>
<td>74</td>
<td>93</td>
<td>439</td>
<td>441</td>
</tr>
<tr>
<td>Nets</td>
<td>17</td>
<td>91</td>
<td>114</td>
<td>480</td>
<td>441</td>
</tr>
<tr>
<td>Time</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>Gain</td>
<td>26</td>
<td>225</td>
<td>189</td>
<td>355</td>
<td>420</td>
</tr>
</tbody>
</table>

9.1 Cost Gain

I have tried to analyse the algorithm with various sizes of ckt. I observe that as the no:of cells increase the cost gain also increase.
9.2 Speed

The Speed required is calculated by taking smaller ckt's first and then bigger ckt's. The time required for the algorithm increases as the size of the ckt increases.

Let's assume the Pentium 4 processor has 55 million transistors and each transistor takes 1 cell then in that case my algorithm took 21 seconds for 441 cells. So for 55x10^6 cells it will take 30 days.

Consider this: \( \frac{55000000}{x} = \frac{441}{21} \) so

\[ x = \frac{55000000 \times 21}{441} \]

\[ x = 2619047.6 \text{ seconds} \]

\[ x = 727 \text{ hours} \]

\[ x = 30 \text{ days} \]