

# **Simulated Annealing Homework**

T. Nathan Mundhenk and Laurent Itti  
University of Southern California  
Department of Computer Science

Copyright (c) 2006 T. Nathan Mundhenk, Laurent Itti

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

## CS561 Homework 2

**Due: By class Thursday, February 16, 2006**

### Guidelines:

This assignment has a written and programming component. You can see how much each part of the assignment is worth by the percentage next to it. For the written part, please turn in typewritten answers. You are not English majors, however blatant spelling and grammatical errors may cost you points, so be sure and run the spell check at least once. Answers on the written part will require justification and a simple yes/no answer will almost certainly garner you little or no points. In general, it is safer to write longer answers than shorter ones, but stay focused as a long but off-topic answer will not work either. This way, we can discern your train of thoughts better and grant partial credit for a wrong answer, if you were on the right track. Graphs must be neat and tidy. Hand-drawn graphs are OK, but computer-drawn graphs (e.g., made with Adobe Illustrator or Microsoft Power Point) are preferred. If you draw a graph by hand, be sure and use a straight edge (ruler) so it looks neat.

For the programming part, you will be provided sample inputs and at least one sample output. Since each homework is checked via an automated Perl script, your output should match the example format *exactly*. Failure to do so will most certainly cost some points. Since the output format is simple and there is an example on the web, this should not be a problem. Additionally, if your code generates a large number of warnings during compilation, you may lose points, so try and eliminate compile-time warnings. Additionally, your code should be well documented. If something goes wrong during compile and grading, if the fix proves easy, the amount of points lost will be far less. As such, documentation makes fixing easier, so it is to your advantage to do so.

You will be provided with a stub Perl script called “stubby.pl”, which works like the grading Perl script. You can run this script to make sure that your project will work properly. Thus, it is expected that your project will run through the grading Perl script without problems. Pedantically, we assert it will cost you points if your code does not run on the Perl script correctly. You will be able to tell if your output is correct if stubby shows each of the lines from your program output is exactly the same as from the comparison file which shows what output you should be getting.

To run stubby, copy it to your code directory along with the input, output and comparison files. Be sure to be in your code directory then type “perl stubby.pl”. The script is loaded with all sorts of output and feedback, so you should be able to see what it is doing if for some reason it isn’t working for you. If you don’t understand the script and want to know more about Perl, go to <http://www.perl.org/>.

Small amounts of extra credit (not more than 10% total) are given for all sorts of things. So long as you meet the base homework requirements anything creative, fun, interesting or outright cool

will most likely earn you extra credit. What is cool and earns you extra credit is somewhat subjective and bound to the whim of the grader.

### **Handing in the Assignment:**

Since the class is very large it is important to hand in your homework as directed. *The homework is due before class on the due date shown.* There are two parts of the homework, which you must hand in. These are:

- (A) Your code tar/gzipped in **one** file. Do not send the binaries. - This should just include your uncompiled code and a readme file called readme.txt. When I run tar/gzip to uncompress your file, it should uncompress into its own directory. To keep things uniform, do not use bzip2. I should then be able to run my stubby Perl script and grade your code. See below on how to tar/gzip your code.
- (B) Your written part on paper. *Iff* you are a DEN student, you may email your written part to cs561. Otherwise, you must submit a paper copy either in class or in a drop-box in front of my office in HNB (NOTE: the HNB building closes at 5:00pm). Late submissions will be noted. Be sure your homework is stapled together and is generally tidy. Electronic submissions of the written part from non-DEN students will not be accepted.

Be sure that your name is *clearly visible* on all material you hand in. To hand in your code you can compress it with tar/gzip with a command like:

```
“tar -czvf my.name.hw2.code.tgz my.name.hw2.code”
```

You might also try:

```
“tar -cvf my.name.hw2.code.tar my.name.hw2.code” then type “gzip my.name.hw2.code.tar”
```

This will compress the contents of the directory named “my.name.hw2.code” into a single file my.name.hw2.code.tgz. If you need more info on this, try “man tar”.

### **Electronic Submission of code:**

You need to submit the assignment electronically using the EXACT following command FROM YOUR ALUDRA ACCOUNT.

```
submit -user csci561 -tag hw2 my.name.hw2.code.tgz
```

Be sure and substitute “my.name” with your own name. The submit command will immediately respond with a SUCCEEDED if your submission of file "my.name.hw2.code.tgz" is successful. That will be your means to know that your homework has reached the right place. Your submissions will be time stamped, so we will know the exact time when you made the submission.

### **DEN students using electronic submission of the written part:**

For DEN students handing in the written part electronically, if you use Word files, please compress the file using .zip or .rar.. Don't worry about compressing Acrobat documents since they are already compressed. Thus, I should see two files if you are doing an all-electronic submission, one with a name like bob.bobbinopolis.hw2.written.zip and the other with a name like bob.bobbinopolis.hw2.code.tgz. If you send me a file with a name like hw2.zip, I will put a hex on your credit cards. Of course this does not apply to your .cpp/.h files which should have names that work with the stubby Perl script. Also, be sure your name is on the documents you hand in since we print them up to grade them and it can be hard keeping documents matched. Please make sure all images and material for the written part are collated into one single file. Thus, *do not* send a word document and 30 image files, be sure and place them inside the word document itself.

### **What exactly is Stubby?**

*This is very important.* We use a Perl script to automate the grading process. Stubby is a Perl script we give to you so that you can check and make sure that your project conforms to specifications. That is, you can use Stubby to make sure that your assignment when handed in will run with our grading script. Thus, we have our own grading script *like* stubby, which we use to grade your project. By checking to make sure your project works with stubby, you make sure that your project will work with our automated grading script.

It is important to note that we will not use your version of stubby. We have our own script. As such, if you edit stubby to work with your code and not the other way around, this will not be very helpful. Additionally, since there are so many projects to grade, it is imperative that your program work with the grading Perl script. *If your program does not work with the grading Perl script you will lose points.*

### Question 1 (20%)

A major video game manufacture has hired you to figure out how to minimize the amount of sleep their programmers get by adjusting different items in the workspace. For instance, you could:

- (1) Turn the amount of caffeine added to their coffee up or down.
- (2) Adjust the lights to be brighter or dimmer.
- (3) Adjust the average pay up or down.
- (4) Make their seats harder or softer.
- (5) Turn up or down the volume on the corporate Muzak.
- (6) Reduce or increase trace amounts of Xylene in the air.

You can tell how much sleep each programmer is getting based on feedback from a microchip each one has been implanted with in their head. You then have a digital readout with the average amount of sleep for all the programmers.

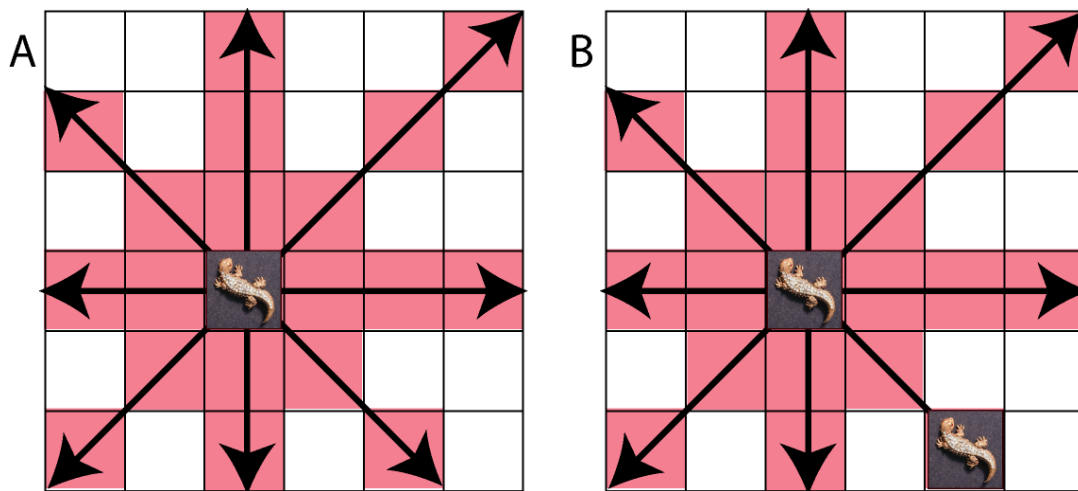
(Questions 1-4, 5 points each)

- (1) Thinking about methods we have studied so far, what would you do to find the optimal adjustments for the workplace items in order to minimize sleep? Describe how you would implement such a method in this situation.
- (2) The video game manufacture asks you if you can guarantee that you will find the absolutely optimal workspace settings such that programmers get the least amount of sleep possible. What do you tell them?
- (3) You notice that one of your former TA's from CS 561 now works at the video game manufacturer and you want to be nice to him. Could you use the same procedure to maximize the amount of sleep each programmer gets? Explain.
- (4) It turns out one of the officers at the software manufacturer is a real big fan of Charles Darwin. He asks you to figure out a way to do the sleep minimization using a genetic algorithm. How might you change the experiment to use a genetic algorithm? You may assume there are enough programmers and resources at your disposal to carry out such an optimization.
- (5) Allowing you to assume this last question is worth no points, is the plight of each programmer more Orwellian or Kafkaesque?

**Question 2 (Coding 60%, Questions/Experiments 20%):**

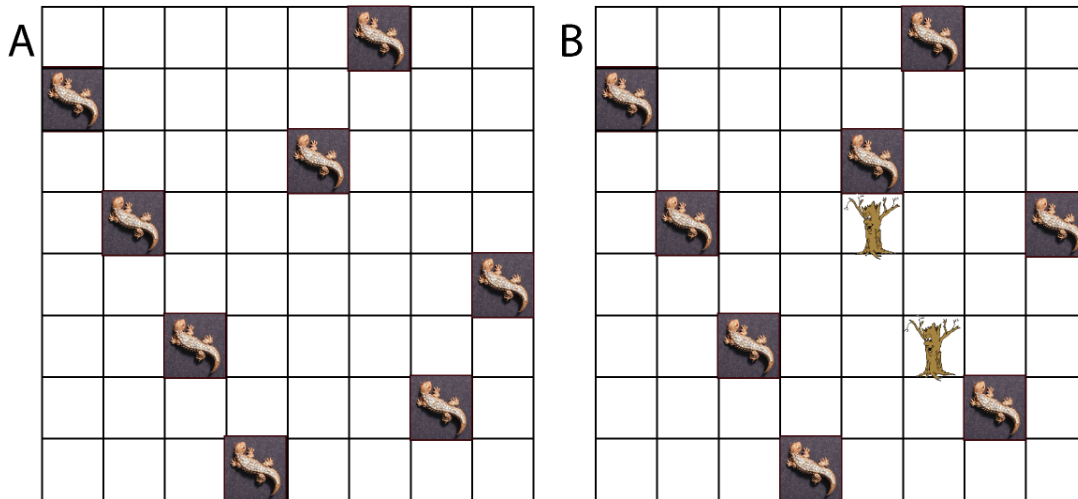
You are a zookeeper in the reptile house. One of your rare South Pacific Tufted Wizzo lizards (*Tufticus Wizzocus*) has just had several babies. Your job is to find a place to put each baby lizzard in a nursery. However, there is a catch, the baby lizzards have very long tongues. A baby lizzard can shoot out its tongue and eat any other baby lizzard before you have time to save it. As such, you want to make sure that no baby lizzard can eat another baby lizzard in the nursery (burp).

For each baby lizzard, you can place them in one spot on a grid. From there, they can shoot out their tongue up, down, left, right and diagonally as well. Their tongues are very long and can reach to the edge of the nursery from any location. Figure 1 shows in what ways a baby lizzard can eat another.



**Figure 1** (A) the baby lizzard can attack any other lizzard in a red square. Thus it can be seen that a baby lizzard can eat another lizzard to its top, bottom, left right or diagonal. (B) In this setup both lizzards can eat each other.

In addition to baby lizzards, your nursery has some trees planted in it. Your lizzards cannot shoot their tongues through the trees nor can you move a lizzard into the same place as a tree. As such, a tree will block any lizzard from eating another lizzard if it is in the path. Additionally, the tree will block you from moving the lizzard to that location. Figure 2 shows some different valid arrangements of lizzards



**Figure 2** Both nurseries have valid arrangements of baby lizards such that they cannot eat one another. (A) with no trees, no lizard is in a position to eat another lizard. (B) Two trees are introduced such that the lizard in the last column cannot eat the lizard in the second or fourth column.

You will write a program in c++ that will take in an input file that has an arrangement of lizards and trees and will output a new arrangement of lizards (no you can't move the trees) such that no baby lizard can eat another one. You will be required to create a program in GNU C++ (g++) that finds the solution. You may either use the g++ version on Aludra or g++ versions 3.3 to 3.4.1 on Linux. Be sure and *clearly* note which one you used so I know which machine to compile it on. To find the solution you will use **simulated annealing**. Additionally, you will be required to run several different test files and give some analysis of your running.

**Input:** Your input will start with a single number, n, followed by the n x n nursery. It will have a 0 where there is nothing, a 1 where there is a lizard and a 2 where there is a tree. You will output a new file exactly like the first (except for one extra line, see below), but with the correct lizard positions. So for instance, an input file arranged like figure 2b would look like:

```

8
0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 1 0 0 2 0 0 1
0 0 0 0 0 0 0 0
0 0 1 0 0 2 0 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0

```

The name of your input file will be specified as the first command line argument in your program.

**Output:** At command line an output filename will be specified as the second argument in your program. The output is the last state of your program. If you found a solution, the output file should contain that solution. Else if you did not find a solution, output the last state. You will specify that you found a solution by placing a 1 at the last line of your output. If you found no solution, place a 0. Thus, if your output looks like figure 2a, your output file will look like this:

```

8
0      0      0      0      0      1      0      0
1      0      0      0      0      0      0      0
0      0      0      0      1      0      0      0
0      1      0      0      0      0      0      0
0      0      0      0      0      0      0      1
0      0      1      0      0      0      0      0
0      0      0      0      0      0      1      0
0      0      0      1      0      0      0      0
1

```

(note that the output example shown here does not correspond to the above input example; the two are not related, since, again, you cannot move (or remove) any tree that may be present).

### Experiments:

Your program will be run on several different input examples and each time should output the solution if one exists. Additionally, keep track of the amount of time your program takes to run, in number of steps. So, you will need to keep a counter that increments by one every time you move one of your baby lizards. Additionally, you will need to be able to change the temperature schedule to find the one that works best. When you find the optimal schedule, fix that as the default for the code you hand in.

### Note on time restrictions:

If your program takes longer than 30 seconds on any of the example problems you will lose points for it. So please try to keep run time within acceptable bounds.

### Questions:

- (1) (2 points) Why is simulated annealing a good algorithm for this problem? (think about complexity and other considerations)
- (2) (6 points) What would be reasonable temperature decrease schedules to use for this problem? To answer this question you will need to do some search on the web to find which schedules other people have typically been using for various problems solved using simulated annealing. Tell us about at least three different possible schedules. Include equations and function graphs for each schedule as well as the URL from which you found it.
- (3) (10 points) Experiment with the different temperature schedules you found and conduct a comparative analysis, running your algorithm on at least 25 different problems of various

sizes each time (that is, create inputs for 25 problems of your choice; then run your program on that same set but each time using a different temperature decrease schedule. It is okay if your program has to be recompiled to select which schedule to use. When you send us your code, just have the best schedule as your default. Motivate your answer with a table showing some measure of performance on your 25 test problems (for example, steps taken) for your different schedules. Like in (2), you should test at least 3 different schedules. Include a conclusion on which schedule seems superior with supporting statistics and illustrations.

- (4) (2 points) In addition to using simulated annealing, could you also have used genetic algorithms to solve this problem? If so, how would you have done it (just explain, you do not have to code), if not, then why?