Approaching P=NP: Can Soap Bubbles Solve The Steiner Tree Problem In Polynomial Time?

Long Ouyang

June 14, 2006

Abstract

Computation theory folklore has it that soap films calculate Steiner trees. If true, this would imply P=NP. However, no rigorous physical experimentation has been performed on the matter. Indeed, such experimentation would be difficult to establish standards for. Furthermore, popular opinion has it that since soap films are classical physics objects [2], this verifies a physical version of the Church-Turing thesis, which must imply P=NP. Careful analysis reveals that such a line of reasoning cannot be used in a formal proof, since, at present, the Church-Turing thesis cannot be proven. Research in previous fields [3] [4], suggests that using physical systems as analog computers is entirely possible. However, critics state that while physical systems can relax to local minima, there is no guarantee that they will relax to global minima [1]. To determine the plausibility of the method in question, I will construct a computational model of soap films, to examine, with reproducible results, the action of soap films in forming Steiner trees.

1 Introduction

1.1 P and NP

In computer science, it is often convenient to formulate questions as decision problems. A decision problem is a yes/no question. Concerning decision problems, there are two complexity classes that have generated widespread interest. These are P, the class containing decision problems which can be solved by a deterministic Turing machine in polynomial time, and NP, the class containing decision problems which can be solved by a *non-deterministic* Turing machine in polynomial time. By polynomial time, we mean that, given some input of size n for a decision problem, the run time of the correct algorithm is bounded by some function, cn^k, where c and k are constants.

More simply put, P contains problems whose answers can be determined in polynomial time and NP contains problems whose answers can be verified in polynomial time using a deterministic Turing machine, which is what modern programming languages produce.

A great question in complexity theory asks, is P=NP? In other words, are problems we previously thought were rather difficult actually relatively easy? Many researchers have tackled the question, though to date, no one has produced an answer accepted by the general field of computer scientists and mathematicians. Great attention has been given to problems in NP, and the most difficult problems have been organized into a subset of NP, NP-Complete. A discussion of the rigorous definition of NP-Completeness is beyond the scope of this document; suffice it to say that NP-Complete problems are the ones least likely to be proven in P.

1.2 The Steiner Tree Problem

A wide number of industries must produce the solutions to problems in NP. Solving NP problems has applications in cryptography, natural language processing, scheduling theory, logic, and more. We will concern ourselves with the Steiner Tree Problem, also known as STEINER.

STEINER asks: given a weighted graph $G(V, E, w)$, where V is the set of nodes, E is the set of edges, and w is the set of weights, and a set of vertices $S \subseteq V$, find the subset of G that contains S and has the minimum weight. Simply put, STEINER asks us to find the shortest connected graph of some set of points when we are allowed to add new points.

STEINER is interesting because there is anecdotal evidence for a solution that is in P. Soap films always act to minimize surface area. If we were to dip two glass plates with pegs between them into soap water, the soap would form a Steiner Tree around the pegs. If we take the act of doing such a thing to be the algorithm for STEINER, it is found to run in polynomial time. Obtaining two glass plates runs in constant time, putting in the pegs requires linear time, and soaking the plates in water and taking them out runs in constant time. If we can model this physics digitally, it may be possible to produce a solution for STEINER in P.

2 Background

Research in using physical systems, such as fluids [4], gases [5], and general systems of classical objects [3], as analog computers has been successful. However, there are often subtle design pitfalls that introduce errors into the systems [5]. A common myth is that the Church-Turing thesis, which states that any possible computation can be performed by a Turing machine, implies that physical systems can simulate non-chaotic processes in polynomial time. However, both this verison of the Church-Turing thesis, as well as any physical system that is purported to "always" find a global optimization, must be examined with strict scrutiny. First, the Church-Turing thesis, as postulated by Church, and Turing, says nothing of physical processes. Speculation that classical physical systems are Turing machines, and so can perform computation, and hence fall under P, is largely unfounded. Accounts of physical systems that solve computationally hard problems are often apocryphal and many are in fact demonstrably incorrect [1].

On this topic, numerous dubious sources are easily found, but rigorous experimentation and analysis is sparse. This reflects that the scientific community tends to take the subject somewhat light-heartedly, perhaps given the intractability of the P=NP problem, as well as the extensive difficulties that it produces in other fields. The relative lack of serious formal literature in this area provides the motivation for this project.

3 Development

3.1 Background

The purpose of this project is to model soap films and the way they change to minimize surface area. Water in an open container will form a miniscus, since it is a polar molecule, which exerts slightly attractive forces on other polar molecules. The fact that the top layer of water molecules is not completely surrounded by other water molecules, but rather by air molecules on top produces the signature curve that we see in beakers and test tubes. Surfactants, such as soap, decrease surface tension, which allows fluids to flow more easily.

3.2 Model

The simulation will take the general form of a box, with pegs placed on the inside at arbitrary positions. The box is initially filled with fluid, which will slowly flow out of an opening at the bottom.

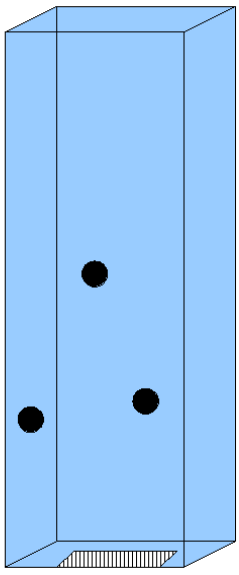

Figure 1: The initial state of the simulation.

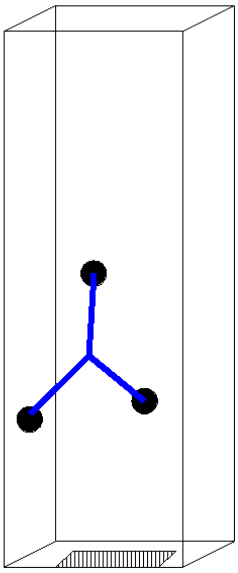

Figure 2: The final state of the simulation.

When the soap has settled, a graph connecting the pegs should be formed.

The surface boundary is calculated by the VOF method and arises as a result of gamma fraction interaction between the fluid and the atmosphere. Presently, the viscosity is modelled by the Bird-Carreau model for non-Newtonian fluids.

3.3 Tools

- OpenFOAM 1.2 - a computational fluid dynamics engine.
- Paraview 2.2 - a data visualization engine.
- GeoSteiner '96 - exact STP solver.

3.4 Software design

Clearly, to verify the computational function of soap film, the simulation must be capable of testing different arrangements of pegs. This necessitates a program to generate input and correct output for the model, for comparison with the results of the simulation. We will call the software module that performs this task the driver. The driver first generates corresponding program input for both OpenFOAM and GeoSteiner. Next, it calls GeoSteiner and OpenFOAM. GeoSteiner output is converted to a graphical PostScript file, and OpenFOAM results are verified using Paraview.

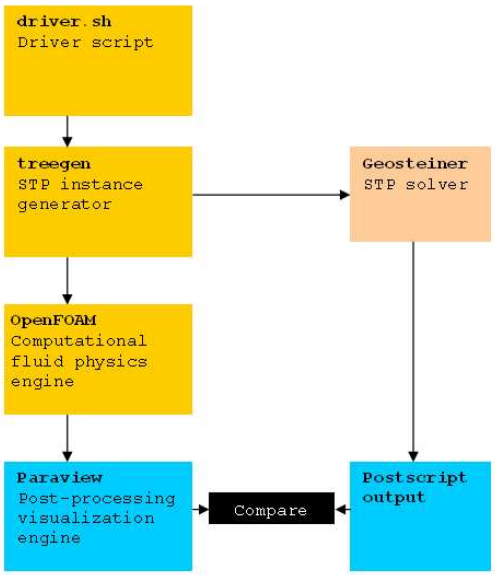

Figure 3: Software flow.

4 Results

A bug was discovered in the OpenFOAM meshing utilities that prevented the mesh operation from being integrated with the fluid simulation. This precluded the possibility of any full testing of STP-satisfiability. A new version of OpenFOAM was released that fixed this bug, but this occurred too late in the development cycle.

5 Conclusion

Significant headway was made toward a fully function simulation. An extensible framework was developed for future research, which will undoubtedly focus on creating a functional experimental setup.

6 Improvement

- Move to OpenFOAM 1.3, to create a fully functional model.
- Implement object oriented programming techniques, which are better suited for handling objects and relationships such as the ones in this project.
- Modify the physics simulation to use Newtonian fluid transport equations, which better model surfactant-containing solutions.

7 References

- NP-complete Problems and Physical Reality, Scott Aaronson
- P=NP, Selmer Bringsjord and Joshua Taylor
- Newtonian Systems, bounded in space, time, mass, and energy, can compute all functions, E.J. Beggs, J.V. Tucker
- Using three-dimensional microfluidic networks for solving computationally hard problems, Chiu et al
- Physical systems for the solution of hard computational problems, Peter Mattsson
- MIT Surface Tension Lecture, John W.M. Bush,
- Official P-NP Problem Description, Stephen Cook
- Soap Bubbles: Their Colors and Forces that Mold them, Charles V. Boys