CS Research Methodology 2011
National University of Singapore CS2309
A/P Lee Wee Sun

Introduction

1 Research

Research is the process of inquiry to discover the unknown. In may involve uncovering the secrets of nature, invention of new applications or improving the efficiency in some processes. Mostly, it is about working out something that is previously *unknown*. It is often exhilarating - imagine discovering some secret of nature, or inventing something that is used by many people all over the world!

Because it involves the unknown, some people are not comfortable doing research. If you are a person who must always know what to expect, a research career may not be for you. However, facing the unknown is inevitable in real life. So, even if a research career is not for you, skills you learn while learning to do research will still come in helpful. You will learn how to ask the right questions, how to go about answering them, and how to validate answers so that they are likely to be true. Moreover, doing research is one of the best ways of achieving competence as it motivates your studies, focuses it and forces you to understand and examine a topic from all angles.

Research is closely related to problem solving. Often, research is done because a particular problem needs to be solved. A good researcher needs to have good problem solving skills. However, problem solving skills are not enough. A large part of research involves formulating the problem. This is possibly the most difficult part of research - figuring out the right problem to solve so that the outcome of the research is impactful, and formulating the problem such that it gives useful insights or outcomes but at the same time is likely to be solvable. The world is full of problems; you need to choose the ones that matter if they are solved; you also need to choose ones that are likely to be solvable given the resources you have. Experience plays a large part here.

So, what makes a good researcher? First of all, you also have to have training or knowledge in the right area. You also have to be fairly intelligent. It is difficult to research a topic if you cannot understand it. However, it is not neccessary to be a genius. There are many ways to success; persistence and hard work often plays an equal or more important role in successful research. Like in most things, you are also more likely to be successful if you enjoy doing research. You should be motivated - good researchers are often driven; they often think about the same thing for days; they work late and on weekends. However, they usually find pleasure in doing that. They have a curiosity to find out how things work; they are dissatisfied with how things currently work and have a desire to make it work better, to improve it, and make it neater in some ways. And, as mentioned earlier, they are able to withstand ambiguity of not knowing the right answer and of having to change their hypotheses when they do not agree with evidence.

1.1 Great Research

How do you do great research, research of such importance that it often lead to a Nobel prize or a Turing award?[1]

To do great work, first of all, you need to work on important problems. Not impossible problems, but problems that will have impact if solved. Great scientists know the important problems in their field. They always keep them in their minds. When they learn some new ideas or techniques, they always try attacking these problems with the methods. And if the line of attack is promising, they go after it with great urgency. You don't have to be successful many times to do great science. This is what luck means. Fortune favours the prepared mind!

How do you get to know the important problems, or the techniques that are likely to be useful for attacking important problems. Of course, you need to work hard. As Hamming said [4]:

> *Given two people of approximately the same ability and one person who works ten percent more than the other, the latter will more than twice outproduce the former. The more you know, the more you learn; the more you learn, the more you can do; the more you can do, the more the opportunity - it is very much like compound interest.*

Knowing more is not enough. You have to be greatly committed to the problem you are trying to solve. Great commitment often means spending a lot of time thinking about the problem. We still do not know how our brains work but creativity appears to come mostly from the subconscious. You may work hard on a problem and not be successful only to have the answer in your head after a night's sleep. However, this usually only happens after you have spent days or nights thinking hard about the problem. If you are not committed to your problem, your subconscious will just go off and do other things as well!

You also need courage - to be committed to solving a problem when it gets difficult. Great work is often work that is different; otherwise others would have done it! Work that is different often require more evidence to convince others. Such work often have a higher entry barrier, require more persistence before success can be achieved. Consider, for example, the relational database [2]. Using the relational model introduced a whole layer of abstraction between the application and the underlying computing infrastructure. Instead of accessing the data directly, a declarative query is made; the query needs to be further optimized, hopefully automatically, in order to access the data in reasonable time. The benefit is the data independence, where the application does not need to know about how the data is represented physically. But is the overhead worth it? Relational database has been very successful, but this could not have been clear when it was introduced. In contrast, in the 1980s, a large effort was made to push logic programming as the basis for future computing in the Fifth Generation Computer System project. However, on the whole, this was not successful as the overhead was large and logic programming was not adequate for most of the applications it was envisioned for. Research in inherently risky and great commitment and courage is often required for its success.

Finally, you cannot work on only important problems. Otherwise, you may end up being very unproductive. You should ask yourself what the important problems are, and keep them in mind

[1]A lot of material from this section is taken from [4].

all the time. Quite often you will work on related problems that will take you part of the way to solving important problems, or shed some light on how to attack those problems. And whatever problem that you are working on, you should always ask yourself: how do I generalize what I learn from this, so that it can be applied to a larger class of problems? A technique that works on only one problem may be somewhat useful, but with a bit of effort, it can often be made to apply to a larger class of problem - and that may turn out to be important!

2 Types of CS Research

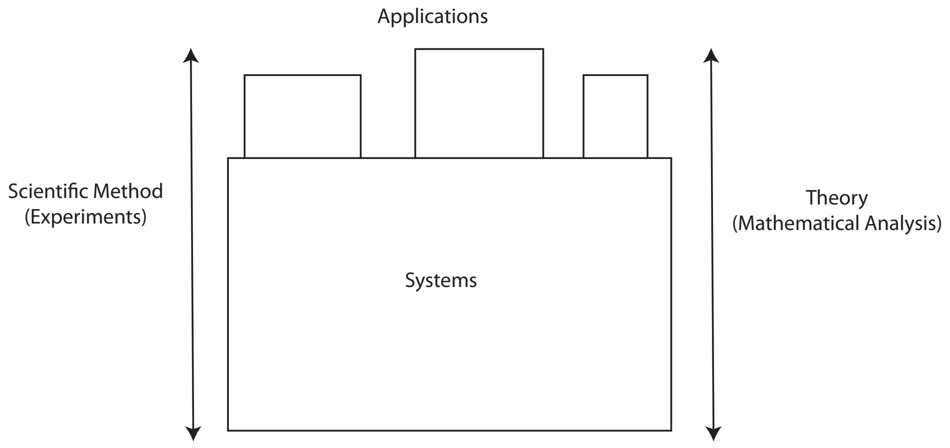
Applications
Scientific Method (Experiments)
Systems
Theory (Mathematical Analysis)

Figure 1: Computer science research.

For convenience, we will organize the description of computer science research around three categories: theory, systems, and applications. The reader should bear in mind that many works span the different categories. Computer science is distinguished from many areas by the fact that it provides a general purpose tool that can be used to solve almost any problem. Systems research is concerned with the topic of how to build such tools so that they are useful and effective. Applications research studies how computing can be used as a tool for solving problems in many specific areas. Theory research applies to both systems and applications. One of the roots of computer science is mathematics, and theory work is essentially the use of mathematics to understand computing. The other main research methodology is the scientific method, where hypotheses are verified experimentally; this is applied in both application and systems research as well.

Besides the breadth of computer science (a consequence of it being a general purpose tool), automation of a process by computers also results in it being scaled to much larger problems. As a result, computer scientists are often concerned with studying how things scale to larger and more complex problems.

2.1 Theory

In a theoretical work, a computer scientist usually abstracts a formal model from a real-world phenomenon. Mathematical analysis is then performed to prove theorems that are useful for understanding the phenomenon. A theoretical work typically provides insights. These insights are often helpful in designing algorithms or architectural principles for practical works.

A classic example is the abstraction of a Turing machine as a model for human computation [7]. Turing was concerned with understanding what computation is and used the Turing machine as a model that he assumed can do everything a human can do when computing. Based on this model, he was able to show that there are problems that computers cannot solved. In doing so, he gives a negative answer to the Entscheidungsproblem. The Entscheidungsproblem was posed by the great mathematician, David Hilbert, in 1928 as a challenge. It asked for an algorithm that takes a formal language and a mathematical statement in that language and outputs whether the statement is true or false.

Theoretical works often try to characterize what is feasible or the best possible. For example, it can be shown that the running time for comparison-based sorting is $\Theta(n \log n)$. The result of such analysis can be used to evaluate and compare different methods, e.g. to claim that in the worst case, merge sort with a running time of $\Theta(n \log n)$ is better than insertion sort with a running time of $\Theta(n^2)$.

2.2 Systems

While theory methods have their roots in mathematics, systems research requires a lot of engineering. Typically, a research group builds a large system to realize a vision. For example, based on the relational model of Codd [2], several groups, including the System R group at IBM research and Ingres group at Berkeley built relational database systems and demonstrated that the idea can be made into useful practical systems.

Systems research often requires large groups and research often depends on parts built by others. To make the computer systems work, many useful techniques are often invented: virtual memory, garbage collection, packet networks, transactions, graphical user interfaces, etc. The resulting systems can often be commercialized, e.g. the Google search engine. Often, they becomes part of the computing infrastructure, such as the Unix operating system or the Internet (result of ARPANET research).

Systems research are often validated theoretically through analysis or empirically through measurements. Interestingly, the system itself is often the validation, i.e. the existence of a useful system that fulfill real user needs. Case studies are often done by studying a successful system to understand the factors behind the success. Experience from previous systems is used to build better future systems.

2.3 Applications

Computers can do virtually anything, so the application areas are diverse. Logistics, computer graphics, vision, natural language processing, information retrieval, computer games, robotics and computational biology are some of the common application areas.

Often, research contribution in an application area is an abstraction that models important aspects of a problem or an algorithm that does something important in the application area. Models with computationally tractable algorithms are particularly useful. For example, in computational biology, finding similar genes are often modeled as string comparison problems [5]. Efficient algorithms for string comparison can then be applied to create a useful tool for biologists. Similarly, speech recognition is often done using hidden Markov models, a type of probabilistic state machine. Hidden Markov models have efficient inference algorithms contributing to the success of the model.

A key aspect of applications research is understanding the important properties of the domain. Often this requires substantial knowledge of the domain or collaboration with experts in the domain. Validation techniques in application areas are similar to those in other areas of science. Theoretical analysis reveals insights into properties of the models or algorithms and experimental work shows how the methods are likely to perform on realistic instances of the problems.

2.4 Scientific Method

Computer science is a combination of science, mathematics and engineering. So, it is not surprising that the scientific method is used in both the applications and systems areas to validate findings. The scientific method has been developed over centuries to validate scientific results using observations. In the scientific method, typically, a hypothesis is first made about some aspect of the problem under study. An experiment is then performed to gather observations on the hypothesis. Statistical methods are then used to reject or accept the hypothesis based on the observations. Since science is a continuing endeavour, the outcome is often used to further improve the scientific model under consideration.

3 Selecting a Research Topic

How do you go about finding a good research topic to work on? If you work for a company and your boss gives you a problem that must be solved, then you don't really have a choice. Or do you? You should always ask about the purpose of solving the problem. Then ask whether solving the problem as formulated achieves the aim. Would solving a different problem would achieve the objective better?

However, let's assume that you have the freedom to select the topic and your aim is to achieve research impact. First of all, you should always work on something that you are interested in. Almost all research encounter difficulties along the way, and if you are not interested, it is difficult to maintain the required effort when things get difficult - and perseverance is often required to achieve good results.

You should consider the impact of the research that you are doing or are going to do. What is the outcome if you are successful?

- Would it improve the lives of people? For example, research on ARPANET turned into the Internet and changed how we live.
- Would it generate wealth, for yourselves or for others? For example, Google was an academic search engine, but also turned into a multi-billion dollar business.
- Would it improve our understanding of the natural world, or of the mathematical world? For example, the theory of NP-completeness have incredible consequences on what is mathematically or even physically possible.
- Would it allow things that cannot previously be done to become possible? For example, invention of public key cryptography allows secret communication between parties without prior exchange of keys.
- Would it make things 10 times, or 100 times better? For example, speedup due to algorithmic improvements in linear programming has exceeded speedup due to Moore's Law since the invention of the interior point methods in the 1980s.

In thinking about potential impact, you can often gain insights by looking at the technology trend. Moore's law, which predicts an exponential increase in computing power, is an interesting example. For many years, parallel computing systems have difficulty gaining traction, except for specialized use, because exponential increase in single processor performance made them less cost effective. For example, the failure rate of parallel computer companies is 100%: Convex, Encore, MasPar, NCUBE, Kendall Square Research, Sequent, (Silicon Graphics), Transputer, Thinking Machines, etc. Another example is the Fifth Generation Computer Systems project in the 1980s which was partially dedicated to developing highly parallel computer architectures. Part of the reason it was not successful was that lack of cost effectiveness of such architectures in comparison to the exponentially improving single processor.

Interestingly, we are at an important crossroad of this particular trend. Single processors are getting too hot to keep improving at the same exponential rate. While Moore's law still holds for the doubling of transistor density in integrated circuits, the microprocessor industry has switched to increasing the number of processor cores instead of increasing processor speed to handle the heating and power consumption problem. This means that improved performance in future will have to come from parallelism instead of increasing clock speed. So looking at the trend, perhaps, parallel computing and power aware computing are currently good research topics.

While observing the trends in computer systems can provide useful hints for what is worthwhile, you should always also examine the needs of the user. One reason for the failure of parallel computing to catch on is the lack of applications that needs them. This may be a chicken and egg problem - as parallel computing becomes more cost effective, more applications may start taking advantage of them. However, in order to develop cost effective systems, designers need to examine the characteristics of potential applications and design their systems to exploit recurring patterns of these characteristics. An interesting study of patterns in parallelisms that can be exploited can be found in [1].

Another factor to consider is your relative strength. Are you good at that particular topic? Do you have skills that others do not have? Can you do it better than others? What is your secret weapon?

Having said all that, I would like to re-emphasize that you should do something that you are interested in. If you are not having fun (at least some of the time), you are unlikely to be able to sustain the research. You should certainly exploit potentially impactful topics, particularly if you have advantages (recall that great researchers keep important problems in their minds and exploit them whenever they encounter a possible line of attack). However, you should not forget to explore other things at other times. Much of what you learn can become useful in unexpected circumstances. A lot of new ideas come about because you have deep understanding of an area and can see the connections between the area and new problems that you are trying to solve.

3.1 Getting Started

Having selected an topic that you like, how do you then get started. You need to gain a lot of knowledge about the area through reading and perhaps taking courses. But it is usually a good idea to get your hands dirty and try to do some research as quickly as possible. Even if the research does not work out, you will learn a lot. Sometimes the ideas would lead to further ideas and even unexpected breakthroughs.

So how do you then select a problem to try? One way is to read research papers, go to the last section, read about the open problems that the authors failed to solve and go solve them. While this may work for some people, it is often not a good idea for many. Problems that others have worked on and failed to solve are often very difficult problems. If you are smart enough, you may be able to solve problems that others have failed to, so it's always worth thinking about the open problems that people write about. An advantage of this is that the work of problem formulation has, to a large extent, been done for you. But beware that the problem may be very difficult.

More often, while reading about the work of others, you may realize that you can extend the ideas, generalize the methods or use the ideas in your own work. This also happens while listening to talks in conferences or while talking to people. The key is to always be thinking about how to extend, generalize or use ideas that you are finding out about. You need to be prepared with a good understanding of important current problems in your area so that you can exploit any new ideas that you learn about. You need to read, go to conferences and talk to people. This is probably also the easiest way to start doing some original research. Look at what someone else is doing and try to extend it a little. Since it is an extension, it is not too hard to do, but it may be enough to extend the state of the art a little[2].

References

[1] K. Asanovic, R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, et al. The landscape of parallel computing research:

[2]Most research works only extend the state of the art a little.

A view from berkeley. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-183*, pages 2006–183, 2006.

[2] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):387, 1970.

[3] R.P. Feynman, R. Leighton, and E. Hutchings. " Surely you're joking, Mr. Feynman!". 1985.

[4] R. Hamming. You and your research. In *Transcription of the Bell Communications Research Colloquium Seminar*, volume 7, 1986.

[5] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[6] Claude E. Shannon. Creative thinking. *Mathematical Science Center, AT&T*, 1993.

[7] A.M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(1):230, 1937.

Appendix: Where do Nobel Prize ideas come from? ☺

You never know where really good ideas will come from. So it's important to explore, do things you enjoy, and understand the basic things really well. But you also should always try to connect up important problems with things that you may already know from your exploration and earlier works. Here's a little story by Richard Feynman [3] on his Nobel Prize idea:

> *... I was in the cafeteria and some guy, fooling around, throws a plate in the air. As the plate went up in the air I saw it wobble, and I noticed the red medallion of Cornell on the plate going around. It was pretty obvious to me that the medallion went around faster than the wobbling.*
>
> *I had nothing to do, so I start to figure out the motion of the rotating plate. I discover that when the angle is very slight, the medallion rotates twice as fast as the wobble rate–two to one. It came out of a complicated equation! Then I thought, "Is there some way I can see in a more fundamental way, by looking at the forces or the dynamics, why it's two to one?"*
>
> *I don't remember how I did it, but I ultimately worked out what the motion of the mass particles is, and how all the accelerations balance to make it come out two to one.*
>
> *I still remember going to Hans Bethe and saying, "Hey, Hans! I noticed something interesting. Here the plate goes around so, and the reason it's two to one is . . ." and I showed him the accelerations.*
>
> *He says, "Feynman, that's pretty interesting, but what's the importance of it? Why are you doing it?"*

"Hah!" I say. "There's no importance whatsoever. I'm just doing it for the fun of it." His reaction didn't discourage me; I had made up my mind I was going to enjoy physics and do whatever I liked.

I went on to work out equations of wobbles. Then I thought about how electron orbits start to move in relativity. Then there's the Dirac Equation in electrodynamics. And then quantum electrodynamics. And before I knew it (it was a very short time) I was "playing"–working, really–with the same old problem that I loved so much, that I had stopped working on when I went to Los Alamos: my thesis-type problems; all those old-fashioned, wonderful things.

It was effortless. It was easy to play with these things. It was like uncorking a bottle: Everything flowed out effortlessly. I almost tried to resist it! There was no importance to what I was doing, but ultimately there was. The diagrams and the whole business that I got the Nobel Prize for came from that piddling around with the wobbling plate.