NSF Science and Technology Center for Scalable Algorithms

Project Summary

Algorithms play a key role in all uses of computers, ranging from the highest level scientific applications to the lowest level design of chips and routers. With the explosion in the size and the complexity of problems that are now being solved, the need for sophisticated algorithms and the theoretical analysis of these algorithms has increased significantly across all domains. Unfortunately the process of incorporating sophisticated algorithms into applications is currently very inefficient in several ways, including:

- an unnecessarily large delay from algorithm design to algorithm application
- an unnecessarily large delay in formalizing real-world problems so they can be analyzed
- algorithms that are optimized for the wrong criteria
- algorithms and techniques that are reinvented many times over across domains.

This problem is due largely to a substantial communication gap between algorithm designers and application implementors. The importance of this problem was argued convincingly in a widely circulated report on directions for theoretical computer science [1]. Reducing this inefficiency could greatly benefit society by accelerating the advancement of many disciplines.

We believe that reducing the gap, and hence the inefficiencies, can be best addressed by an effort on the scale of a center. The purpose of the Center would be to integrate theory with applications across a wide set of domains. Although the NSF and other sources do fund collaboration within particular domains, this work does not share knowledge, tools, data and educational infrastructure across different domains. It is this need for cohesion not just between theory and practice, but also across domains that justifies the need for a center.

The proposed Center will consist of three components: a research component, an educational component, and a data and libraries component. All of these parts will involve collaboration between the Center and a set of industrial partners. The research component will be embodied in a set of problem-oriented explorations. Each exploration will illuminate a specific applied problem or problem area for the purpose of finding algorithmic solutions. The education component will consist of several parts, including developing a set of case studies, developing algorithm tutorials for domain experts, developing new curricula for teaching algorithms to undergraduates, and offering summer programs for high-school students. The data and libraries component will consist of a well-maintained set of useful resources for researchers and implementors.

The Center will involve at least four industrial partners: Akamai, Celera Genomics, Intertrust, and Yahoo. The partners will help the Center in many ways, including supplying problems and data, participating in the problem-oriented explorations, and hosting student interns. The Carnegie Mellon researchers who will be involved in the Center have significant previous experience bridging the gap between theory and applications. Two of them (independently) are recipients of the ACM Kanellakis Theory and Practice award, another has developed a course on the topic, and all of them together organized a workshop on the topic.
NSF Science and Technology Center for
Scalable Algorithms

Guy E. Blelloch (PI), Manuel Blum (Co-PI), Daniel Sleator (Co-PI)
Carnegie Mellon University

1 Motivation

With the explosion in use and connectivity of computers and in the sizes of data sets being collected, sophisticated algorithms\(^1\) are becoming increasingly crucial for a wide range of real-world applications. In fact, sophisticated algorithms already play a central role in a number of domains. Shortest path and interior point methods are used in airline scheduling. String matching, traveling salesman and max-flow algorithms are used in computational biology. Algorithms for graph separators, maximal independent set, Delaunay triangulation, and point location are used in finite element codes for designing airplanes, cars, and space structures. Furthermore, the need for sophisticated algorithms can only be expected to increase in real-world applications as the complexity and size of data increases. This is because experiments and simulations become more difficult to perform as data scales, thereby requiring theoretical analysis to narrow the experimental search space and sophisticated techniques to improve performance. The current and potential benefit to society of using sophisticated algorithms is very large, and includes helping other scientists solve larger problems or existing problems faster, helping improve the design and safety of just about all the equipment and modes of transportation and communication we use, and helping organizations reduce costs by improving schedules and processes.

The transition of algorithms from theory to applications, however, has been slow. It typically takes ten to twenty years for an algorithm to make it from theory into an application, most algorithms never make it, and there are many cases of naive algorithms being used where a better algorithm would surely improve performance. A critical reason for this delay is the large communication gap between algorithm designers and application implementors. The theoretical algorithm designer is often ignorant of the context in which the algorithm might be used and therefore assumes an oversimplified model and/or problem. Similarly the application implementor is often ignorant of the wide selection of potential algorithmic solutions. Although the algorithms in their initial form are often not practical nor relevant, many of them do have core ideas that can be modified for use in an application. Even if an application implementor knows of such an algorithm, however, they are unlikely to understand it well enough to see how it might be modified to fit their problem or be made more efficient in their context. It therefore becomes very easy to dismiss the algorithm. It is this gap in understanding that causes or at least contributes to the delay.

Table 1 shows several examples of algorithms which now play critical roles in applications, but which took ten to twenty years from when they were invented until when they came into common use. In every case the delay time can be largely attributed to the gap between algorithm designers and application implementors. Consider, for example, the case of

\(^1\)By sophisticated algorithms we mean an algorithm whose techniques and/or analysis has required significant effort to develop. The analysis could be in terms of correctness, performance or quality of the output. In fact the algorithm itself might be quite simple, but the theoretical analysis sophisticated. We also mean to imply by algorithm, a method that solves a somewhat general problem typically applicable across domains, such as sorting or linear programming.
Table 1: A small sampling of algorithms that are in common use today.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Year Invented</th>
<th>Year in common use</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>O(n log n) Delaunay Triangulation</td>
<td>1975</td>
<td>1995</td>
<td>Most meshing packages including those by Fluent, Geomagic, Pointwise, Ansys, FEGS and many others</td>
</tr>
<tr>
<td>Huffman Codes</td>
<td>1952</td>
<td>1984</td>
<td>gzip, bzip, JPEG, MPEG, ...</td>
</tr>
<tr>
<td>Interior Point Methods</td>
<td>1984</td>
<td>1994</td>
<td>Most linear programming packages, including CPLEX and IBM OSL, the two that are most widely used</td>
</tr>
<tr>
<td>Model Checking</td>
<td>1981</td>
<td>1996</td>
<td>Intel chip verification</td>
</tr>
<tr>
<td>RSA</td>
<td>1978</td>
<td>1993</td>
<td>Most public-key systems, including all major web browsers.</td>
</tr>
<tr>
<td>Splay Trees</td>
<td>1985</td>
<td>1995</td>
<td>Microsoft NT, Lucent Concentrators, Fore System’s routers, gcc, Linux, ...</td>
</tr>
<tr>
<td>Strassen Matrix Multiply</td>
<td>1969</td>
<td>1990</td>
<td>SCILIB, IBM ESSL — used in many scientific applications</td>
</tr>
</tbody>
</table>

Strassen’s Matrix Multiply [16]—a recursive algorithm that multiplies two $n \times n$ matrices in $O(n^{2.81})$ steps instead of the $O(n^3)$ taken by the standard algorithms. For many years people dismissed Strassen’s algorithm as completely impractical (see, for example, [13]). This was largely due to the lack of sufficient experimental work and lack of understanding of how the algorithm might be modified to make it efficient. In particular there are several issues having to do with space, cache-effects, numerical stability, and when to stop the recursion that had not been studied. These issues were studied in a sequence of papers in the past decade [2, 5, 9, 18] (more than 20 years after it was invented), and now Strassen’s algorithm appears in several libraries including SCILIB, VECLIB, and ESSL, both as a matrix multiply and as a component of linear systems solvers. These libraries are used to solve many large scientific problems. Similar stories can be told for the deployment of interior-point methods [10], where sparse matrix methods along with other improvements were needed to make them efficient [7], model checking [4], where binary-decision diagrams were needed to avoid having to represent the whole state space [3], Huffman codes [8], where efficient techniques were needed to look up codewords quickly [14], and the Kernighan-Lin graph separator algorithm [11], where more efficient internal data structures were required [6].

Table 1 is just a small sampling of a much longer list of examples depicting the lag between algorithm discovery and algorithm deployment. The cumulative cost to society of these combined delays is hard to estimate, but is surely very large. For example Subramaniam et. al. [17] say that Delta Airlines saved $300/million a year in costs by switching to more advanced fleet assignment algorithms. This switch was made possible by deploying much more efficient linear-programming algorithms, which had been developed years earlier. In the remainder of the proposal we discuss how the Carnegie Mellon Center for Scalable Algorithms will reduce this lag time.
2 Why a Center?

There are several ways in which the communication gap between algorithm designers and application implementors can be reduced. One way is to support domain-specific research grants which bring together algorithm designers with domain experts within the context of the a particular problem domain. For example, such an effort could bring together computational geometers and civil engineers to work on earthquake simulation. This interaction could greatly hasten the use of new triangulation algorithms in real-world earthquake simulations, as well as introduce new interesting problems for the computational geometers to work on. NSF already funds several such projects, and many of the PIs involved in the proposed Center have been involved in such research. Our experiences in this joint research has motivated much of the vision outlined in this proposal.

Although such domain-specific projects are very important, they will not, on their own, achieve nearly as much as the Center that we envision. There is much knowledge and infrastructure that can be shared among these domain-specific projects, and this is not happening. As anyone who has engaged in such a collaboration (between a algorithmist and a domain expert) knows, there are many non-obvious problems to that must be overcome. In addition to the time required to develop, implement and experiment with the application, both sides must invest time to learn a new vocabulary, a new way of thinking, and foreign organizational structures. Any shared resources that can be used to streamline such collaborations – such as domain-specific algorithmic tutorials, case studies, libraries of algorithms, workload characterizations, shared experimental work, or workshops – would be extremely useful. We believe that:

1. There is a lot to be learned by looking at previous experience.
2. Many of the improvements for a particular algorithm made in one domain carry over to other domains as well.
3. Much of the code developed in one domain can be used in other domains.
4. Efforts to educate application implementors or algorithm designers can be shared across domains.
5. Many of the techniques used to make algorithms practical in one domain could be used in other domains.
6. Modifying our curriculum to help students understand the application of algorithms, not just their theory, can help across all domains.
7. Data gathered in one domain, and particularly statistical characterizations of that data, can be useful in other domains.

We believe that the center mode of funding is critical for supporting the shared infrastructure described above. There is no motivation for any one domain-specific effort to maintain a library of data, algorithms, source code, and case studies for various domains. Furthermore, there are no direct sources of funding for the creation of such libraries. One might consider taking separate domain-specific libraries and linking them together, however, the problem would be a lack of uniformity, with respect to notation, definitions, formats, etc. It is this need for cohesion not just between theory and practice, but also across domains that justifies the need for a center. To summarize, the scope of the shared infrastructure that we are proposing to create is simply too great to be implemented by anything less than a center.
3 Center Organization

The center will have three main components: a research component, an educational component and a data and library component. The research and data/tools components will involve close collaboration and technology transfer with our industrial partners.

So far we have four industrial partners, Akamai, Celera Genomics, Intertrust, and Yahoo! Inc., and we plan to include more partners over time. The partners will help the center in many ways, including supplying problems, participating in the problem-oriented explorations, supplying data, and being a sight for student interns.

3.1 Research Component

Research in the Algorithms Center will revolve around an overlapping sequence of Problem-Oriented Explorations. The structure of these explorations is unique. Here are some details.

Each exploration will be centered around a specific applied problem or problem area and will last from one to two years. The purpose of the exploration will be to find algorithmic solutions for that problem domain, hence the name “Problem-Oriented Exploration.” Examples of Problem-Oriented Explorations might include:

- Finding algorithms to help astronomers process astrophysics sky surveys.
- Finding algorithms to help the Pittsburgh Supercomputing Center with its Terascale project.
- Helping Akamai find algorithms for distributed massive data processing of Web accesses.
- Helping Yahoo with algorithms for transparent categorization.
- Finding algorithms for new MEMS-based disk scheduling problems.
- Finding algorithms to help with genome sequencing.

A Problem-Oriented Exploration will consist of several components, including visiting faculty and/or postdocs, visits of students or faculty to industry, and at least two workshops. In the first workshop, the participants will gather to hear a description of the problem from the perspective of the application developers, and to ask questions. Approximately six months later, the second workshop will take place. Here again the same participants will gather, but this time to listen to the algorithmists propose various solutions to the problem. During the time between the first and second stages, much interaction will take place, in the form of visits, student internships, etc.. It may be the case that more than two workshops are necessary in the case of some Problem-Oriented Explorations to satisfactorily model and solve the problem.

The participants in a Problem-Oriented Exploration will consist of application developers and algorithm designers. In addition there will also be translators present. A translator is someone fluent in both the language of algorithm designers and the language of application developers, who can therefore translate and interpret, to accelerate the communication process. Translators are very rare, but fortunately the CMU School of Computer Science, which places a strong emphasis on the integration of theory and practice, has several faculty who can serve as translators for some domains.

In order for Problem-Oriented Explorations to succeed, people must be motivated to attend and contribute. The motivations for the application implementors are that they get
the help of algorithmists working on their specific problems. Furthermore, if the application implementor is not located at a university, there is the additional motivation of making contact with a university. The motivation for the algorithmists is many-fold: Algorithmists get ideas for new problems to work on; they get real-world data to analyze; they get a chance to apply their algorithms to real products. Last, but not least, algorithmists will in many cases obtain funding from either the center of the company. In fact we have requested funding as part of this proposal for visiting faculty for this purpose. For all these reasons, we believe that both algorithmists and application implementors will be amply motivated to attend the Problem-Oriented Explorations.

Proposals for topics for Problem-Oriented Explorations will be solicited from both companies and applied researchers at labs and universities. All the proposals will then be reviewed by a committee, which will choose which ones are best suited for Problem-Oriented Explorations.

3.2 Educational Component

The main goals of the education component are the following,

- Make algorithms accessible and enjoyable to a wider audience,
- more rapidly educate domain experts on new algorithms and algorithmic techniques, and
- better educate the theory community on the needs of the domains

To address these goals we propose to do the following:

**Case studies:** We will develop case studies of how algorithms get used in various applications. These case studies will describe both the experience of getting the algorithm into use, as well as details of the final usage of the algorithm. These studies could serve to help application implementors see how algorithms can be used, to help algorithm designers know how to effectively do technology transfer, and to help motivate the usage of algorithms in course work by serving as concrete examples. Many of these case studies will be developed from the problem-oriented exploration, but we will also solicit other case studies from outside people.

**Tutorials:** Develop algorithm tutorials which are aimed at certain domain experts. For example we might have a tutorial on approximation algorithms designed for biologists. Such a tutorial would differ from a generic tutorial on approximation algorithms in that it would use as examples from biology, and be aimed at being much more accessible to biologists. Some of these tutorials could be developed by the permanent members of the center, but we have also requested funding to help pay for external researchers/educators to develop them. The tutorials will consist of video, slides and a written notes and will be made available as part of a library. The hope is that these tutorials could be used as a lecture or two in a course, in conjunction with a conference, or for individuals to take on their own.

**New Approaches to Teaching Algorithms:** Currently most computer science algorithms courses are taught as a collection of algorithms and techniques without tying them to applications that motivate them. This approach leaves students with the impression that the algorithms they have learned, although perhaps very clever and cute, have little relevance to real applications. This in turn promotes the separation of the
theory and systems in the students' minds, and can leave non-theoretical students with little interest in algorithms. We plan to develop courses in which the applications of the algorithms are much more integrated into the course, as at least motivation. As a start we have already developed a graduate course titled “Algorithms in the Real World” (http://www.cs.cmu.edu/~guyb/realworld.html). The emphasis is on maintaining the same level of theoretical vigor while making the students understand that the algorithms have many real-world applications.

**Programs for High School Students:** The Computer Science Department at CMU has developed a summer program for high school students called “Andrew’s Leap”, which is run by Steven Rudich (http://www.cs.cmu.edu/~leap/). We would integrate this program with the center and add a more significant algorithms component.

The educational component will be fully integrated with the research component. For example, the case studies will be largely based on the problem-oriented explorations and will in turn be used in the course development; the tutorials will be used to help in the problem-oriented explorations; and students will be involved in the explorations at the high-school, undergraduate and graduate levels.

### 3.3 Data and Library Component

One of the responsibilities of the proposed Center would be to maintain a repository of data and libraries that will serve as a resource for students, researchers and implementors using and creating algorithms.

**Data Stockroom:** The first part of this is a collection of databases or datasets associated with important problems. A tremendous amount of such data has been collected as part of ongoing research, but much of it is not available, or is difficult to find. Here are examples of the kinds of data that we would make available and why they would be useful.

Database samples: Samples of very-large collections of data could be very useful for researchers to experiment with algorithms. Examples include subgraphs of the worldwide web, parts of a genome database, or parts of the astrophysics sky survey.

Computational traces: Recorded logs of real request sequences are critical for studying algorithms for caching, paging, scheduling, routing, among others.

Problem instances: Collections of problems instances are very important for debugging and experimentally analyzing the performance of algorithms.

The center would be responsible for extracting the data, maintaining it in a common format, and making it easily accessible.

**Algorithmic Stockroom:** The term *Algorithmic Stockroom* was coined in the Karp Report [1]. (See the next section for more discussion of this report.) The Center would carry out this much needed service. To quote from the report:

... we propose the creation of an Algorithmic Stockroom, a distributed repository of well-tested and documented implementations (in reasonably portable source code) of potentially useful algorithms and data structures. This proposal is both more and less ambitious than the LEDA approach. We wish to greatly increase the supply of easily-obtainable algorithm implementations, but in order to speed the process and
allow the incorporation of many already-existing codes, we do not at the start want to
impose strict rules as to algorithmic interfaces or the programming languages used in
the implementations.

Operationally, the Algorithmic Stockroom could consist simply of a well-publicized web
site, with URLs pointing to locations of the relevant source codes, documentation, and
test results, although some local archiving might be desirable. The site would also
contain pointers to locations containing test data, and would serve as a clearing house
for reports on user experience. There would be no a priori restrictions on the types of
problem domains addressed by the algorithms in the stockroom, although one of the jobs
of the Stock-keeper would be to maintain lists of key algorithms not yet incorporated,
so as to help direct the efforts of those who want to participate in the project.

Since this report was written in 1996, a few algorithm collections have been developed,
such as Skiena’s Stonybrook Repository [15], but none of them adequately addressed needs
described in the report. LEDA [12], which is referenced in the quote, is a library of data
types and algorithms developed at the Max Planck Institute at Saarbrucken.

4 Previous Efforts

There have been several previous efforts to generally promote the interaction of the theory
of computing with applications. In addition to our own efforts, outlined in the next section,
these efforts include the Karp Report [1], various contributions from the NSF Science and
Technology Center for Discrete Mathematics and Theoretical Computer Science (DIMACS),
and the development of the LEDA library [12]. Here we briefly relate the proposed center
with the Karp report and DIMACS—LEDA was mentioned in the previous section.

The Karp Report: This report [1] came out of an NSF-sponsored workshop held in 1995
whose goal was to assess the goals and directions of the Theory of Computing. Richard
Karp was the chair of the committee that wrote the report, and the report had input from
most of the theory of computing community. The report strongly emphasized the need
for better interaction between the theory of computing community and applications, and
made 11 recommendations most of which addressed this issue. Several of the components
in the proposed center are strongly motivated by the following recommendations from the
report, (R2) general bridging activity, (R3) implementing and testing of algorithms, (R6)
interactions between theoretical computer scientists and biologists, (R7) application-specific
theory, (R9) the algorithmic stockroom, (R10) visitors of theoreticians to industry, (R11)
exposing theoretical CS students to applications and experimental research.

The report does not address how these efforts should be supported, but we believe that
having a center on algorithms is the most effective way to do it.

DIMACS: NSF funded the Science and Technology Center for Discrete Mathematics and
Theoretical Computer Science (DIMACS) as one of the initial STC Centers in 1989. After
10 years it is no longer funded as an STC. The main difference between DIMACS and
our proposed center is that the main emphasis of DIMACS was on discrete mathematics
and theory with some outreach to applications, while our emphasis is in the interaction of
theory and applications. We believe, however, that some of the efforts made by DIMACS
were quite successful in helping to bridge the gap between theory and applications. We
therefore have incorporated the most successful of these efforts into our model of a center.
An example is our problem-oriented explorations, which are loosely based on the DIMACS
special-years and their associated workshops.
5 People

The Carnegie Mellon researchers who will be involved in the Center have had significant previous experience in the interaction of theory and applications. In fact CMU faculty involved in the proposed Center have won two of the four ACM Kanellakis Theory and Practice Awards. Furthermore the PIs have already done significant exploration in the specific topic of this proposal. For example, three of the investigators co-organized a workshop on “Algorithms in the Real World” that was held in May 2000 (http://www.cs.cmu.edu/~realworld). The workshop was organized as part of a preliminary investigation of the issues discussed in this proposal. The workshop discussants included Richard Karp (Berkeley, Chair of the committee), Kurt Mehlhorn (Max Plank Institute, Saarbrucken, and founder of the LEDA algorithms library), Monika Henzinger (Director of Research at Google Inc.) Eugene Myers, (Vice President of Informatics Research at Celera Genomics) Robert Tarjan, (Princeton, Chief Scientist at Intertrust, and previous co-director of DIMACS) among several others. Guy Blelloch has also developed a course on “Algorithms in the Real World” and has now been teaching it for 4 years (one year at UC Berkeley).

Finally four of the investigators are from areas outside of the computer science department—D. Durand (Biology), A. Frieze (Mathematics), O. Ghattas (Civil Engineering), and R. Ravi (Grad. School of Industrial Administration). All of these investigators have a significant track record of interacting with the PIs on this project or other computer science efforts, and will be excellent at helping to make the bridge between algorithms and applications.

Here we briefly outline the most relevant contributions of each of the senior investigators.

Guy Blelloch has worked on experimentally analyzing many parallel algorithms, and is author of the CRC Handbook on Computing chapter on parallel algorithms. He also developed a new course on “Algorithms in the Real World”.

Avrim Blum is known for his research in algorithms for machine learning, data mining, and combinatorial optimization, and is also developer of the Graphplan planning algorithm, now used as the basis of many AI planning systems.

Lenore Blum pioneered the field of complexity and real computation, which introduced a theory of computation and complexity over an arbitrary ring or field.

Manuel Blum received the ACM Turing Award for his work in the foundation of computational complexity theory and its applications to cryptography and program checking.

Edward Clarke shared the 1998 ACM Kanellakis “Theory and Practice” award for the development of Model Checking algorithms and their applications to circuit verification, such as in the FP unit for the Intel Pentium.

Dannie Durand is a member of the CMU biology department. She has a PhD in computer science, and works in computational molecular biology and genomics, simulation, and experimental analysis of algorithms.

Alan Frieze has done important work on the application of probabilistic ideas in the analysis of algorithms, and received the AMS Fulkerson Prize for his work on estimating the volume of convex bodies.

Omar Ghattas is a member of the Civil Engineering Department and has been co-principal investigator on three National Science Foundation HPCC Projects—the Earthquake Ground Motion Modeling Grand Challenge, the Computer-Assisted Surgery National Challenge, and the Inversion-Based Seismic Modeling KDI projects.
Mor Harchol-Balter is known for her work on creating a modern queueing theory, that doesn’t rely on unrealistic Markovian workload assumptions. She is the winner of the Sigmetrics best paper award for Integrating Theory and Systems, and has co-authored two patents, licensed to Akamai and Cisco, respectively.

John Lafferty is known for his work in language modeling and information retrieval, and is co-developer (along with PI Sleator) of the Link Grammar natural-language parser, now incorporated as one of the SPEC-2000 integer benchmarks.

Bruce Maggs is known for his work on algorithms and analysis of network routing. He was vice president of Akamai Technologies before returning to Carnegie Mellon in the spring of 2000.

Gary Miller has worked on sequential and parallel algorithm design problems in many areas, including geometry, graphs, and number theory.

Ramamoorthi Ravi has worked on approximation algorithms, combinatorial optimization and computational biology.

Steven Rudich has worked in areas of complexity theory, cryptography, combinatorics, and probability.

Daniel Sleator is co-winner of this year’s ACM Kanellakis “Theory and Practice” award for the development of the Splay Tree data structure, used in applications from network routers to the NT operating system.

In addition to those listed above, we expect many other CMU faculty members to be involved in the center. Here are a few of them: Gerrard Cornujols (GSIA), Mike Erdmann (Robotics), Greg Granger (ECE), Paul Heckbert (CS, Graphics), Takeo Kanade (Robotics), Andrew Moore (Robotics), Jeff Peterson (Astrophysics), Michael Trick (GSIA), Noel Walkington (Math), Larry Wasserman (Statistics).

6 Management

The management of the Center will consist of the following key full-time personnel: Director, Co-director, Administrator, Financial Manager, Industrial Liaison, Workshop Coordinator, and Web/Publication Manager. The director will be a tenured faculty member at CMU. The Industrial Liaison will have the job of fostering the involvement of Center’s industrial partners, and finding new industrial partners. The Workshop Coordinator will organize workshops and be in charge of the Center’s public relations.

There will be several committees set up to help govern the Center. Key decisions involving the direction of the Center will be made by an Executive Committee, which will be comprised of members from both Carnegie Mellon and our industrial partners. An External Advisory Board will be comprised of outside researchers in industry and academia, including researchers experienced in running an STC. Its purpose will be to advise and evaluate the Center. The members would be selected before submitting the full proposal. Other committees will carry out a variety of tasks, such as evaluating proposals for workshops, and selecting visitors and postdoctoral fellows. We leave these details to a later stage in the proposal process.

Carnegie Mellon has had several centers of at least the scale of the proposed center. We therefore fully understand the complexity of running such a center and as an institution have had significant experience with the process.
References


