THE *Quest of the* MCM

Conquering the Math Contest in Modeling

Brian Camley Pascal Getreuer Bradley Klingenberg

Every year, the Consortium for Mathematics and its Applications (COMAP) sponsors the Mathematical Contest in Modeling (MCM), an international contest for undergraduates. We will discuss our strategy for developing models, writing the paper, the contest timeline, and team dynamics.

Contents

1 What is the MCM?

In the MCM, three-person teams are given 96 hours to develop mathematical models to solve a real-world problem, evaluate their solution, and write a research paper describing the results. These papers are generally around thirty pages long. The questions are open-ended and over a broad range of topics. Past problems include fingerprint identification, submarine tracking, air traffic control, and velociraptor hunting strategies.

Does your leisure reading include math books? Have you ever programmed a quick game on your TI-86 to stave off boredom? Is the Mathematica vs. MATLAB debate interesting to you? Have you ever considered simulating the growth of grass? Then the MCM is for you.

Even if you aren't an über-nerd, we recommend the contest to you. Why?

- **Practice for a thesis.** The MCM will teach you a lot about organization, clarity, and time management.
- **Reading papers.** Reading previous research is essential to the MCM, and 96 hours of searching for critical information will develop your ability to extract useful bits quickly.
- **Prototyping skills.** You will learn to look at a problem and come up with a fast prototype solution. This is invaluable in programming, math, or science of any sort.
- **Reputation.** If you manage to win, it looks great on a résumé. Several financial companies make a specific point of recruiting MCM winners.

At the beginning of the contest, you are given a choice between two problems[†]. Over past contests, Problem A tends to be "continuous"—problems with continuously varying parameters, especially the modeling of physical phenomena. Problem B tends to be "discrete"—problems like queuing, routing, and scheduling.

The 2006 contest problems are reproduced in Appendix A (p. 22). A useful 15-minute exercise is to read them with your teammates, discuss which one you would hypothetically choose, and brainstorm possible approaches and simplifying assumptions.

[†]MCM and ICM problems are usually presented together. MCM teams must choose between Problem A and Problem B; Problem C is only for ICM teams.

Judging

MCM papers are judged by a panel of mathematicians and math educators. In the first round of judging, only the paper summaries are read. Papers that pass the first round continue to the following rounds, where papers are more carefully read and ranked into several tiers. By percentage[†], the tiers are

- 60% Successful Participant
- 25% Honorable Mention
- 15% Meritorious
- 1%-2% Outstanding

Additionally, two teams receive the Ben Fusaro Prize, recognizing the development of a creative model and a paper which is pleasant to read. Outstanding papers are considered for the SIAM Prize, the MAA Prize, and the INFORMS Prize from the respective societies.

Contest Rules & Logistics

Your team must be registered for the contest by early February. Once the contest begins, you may not add or change a teammate, though you may remove a teammate if necessary. A team may have at most three students, and no student may belong to more than one team.

Papers must be typed and in English. Solution submissions must be paper only; non-paper materials such as computer disks are not accepted. At registration, each team is assigned a control number. The team control number must appear at the top of every page, along with the page number, for example:

Team #321 Page 6 of 13

Other than the control number, the paper must in no way identify the students, the advisor, or the school.

For more detailed contest rules, see

www.comap.com/undergraduate/contests/mcm/instructions.php.

[†]Tier percentages are approximate; actual percentages vary from year to year.

History

The University of Colorado at Boulder has a strong history in the MCM:

2000	Honorable Mention	Jim Barron, Cristy Shannon, John Herman
	Outstanding, INFORMS Prize	Bill Woessner, Rich Younger, Martin Linckw
2001	Honorable Mention	Grant Macklem, Saverio Spagnolie, Tye Rattenbury
	Meritorious	Jim Barron, Jill Kamienski, Olivia Koski
2002	Meritorious	Darin Gillis, Geoff Goehle, Aaron Windfield
	Meritorious	Moorea Brega, Alejandro Cantarero
	Meritorious	James Barron, Jill Kamienski, Olivia Koski
	Outstanding	Kevin Leder, Saverio Spagnolie, Stefan Wild
2003	Honorable Mention	Joe Carrafa, Kimi Kano, Ian Derrington
	Meritorious	Moorea Brega, Alejandro Cantarero, Corry Lee
	Outstanding, SIAM Prize	Darin Gillis, Aaron Windfield, David Lindstone
2004	Honorable Mention	Ian Derrington, Donovan Levinson, Karl Obermeyer
	Honorable Mention	Arian Lalezari, Sarah Macumber, Matt Martin
	Outstanding, MAA Prize	Brian Camley, Brad Klingenberg, Pascal Getreuer
	Outstanding, SIAM Prize	Moorea Brega, Alejandro Cantarero, Corry Lee
2005	Honorable Mention	Brandon Booth, Rachel Danson, Kristopher Tucker
	Meritorious	Thomas Josephson, Edmund Lewis, Laura Waterbury
	Outstanding	Brian Camley, Brad Klingenberg, Pascal Getreuer
2006	Honorable Mention	Christopher-Ian Davis, Ramsey Majzoub, Brennan Dayberry
	Meritorious	Ben Barrow, Thomas Josephson, Laura Waterbury
	Meritorious	Brandon Booth, Rachel Danson, Benjamin Safdi
	Meritorious	Michael Gurshtein, Josh Destree, Edwin Eng
	Outstanding, SIAM Prize, MAA Prize	Brian Camley, Brad Klingenberg, Pascal Getreuer

We hope this trend continues. That said, don't be too disappointed if your team does not win Outstanding. Teams that participate again often progress one tier higher each year. It is unusual for a team to win Outstanding in their first attempt.

2 A Strong Paper

The MCM is in part a contest of communication skills. Ultimately, it is your paper that delivers your ideas and results to the judges, and thus it is as much a presentation as a report. No matter the quality of your research, you must communicate effectively in order for the judges to see it. During the contest, around half of your team's time should be directed toward writing the

paper. Having a well-written paper is nearly as important as solving the problem itself.

Background Research

Your initial research will be critical in framing the problem. The MCM, like any research, begins by understanding the problem and reading previous research. Learn the basics of the problem context: existing models, previous approaches, and especially the nomenclature.

When we solved the 2004 fingerprint problem our first year, we spent a very unproductive first day before we found the concept of "minutiae" in fingerprints. This led us to the central postulate of our paper: two fingerprints are identical if they would be identified as the same person by the FBI. If we had found Stoney and Thornton's paper "A Critical Analysis of Quantitative Fingerprint Individuality Models" on the first day rather than the third, our paper would have been much more thorough. Ten minutes of research can save you a day's work!

! **Don't limit your research to internet sources.** While internet sources are quick and convenient to find, they should primarily be used to lead to scholarly work or peer-reviewed literature. Once you find a paper on topics relevant to your problem, follow its citations and find the journals that often write about these topics. For example, most of the fingerprint literature is in journals of forensic science, traffic studies are in specialist journals and the statistical physics literature, and irrigation is studied in agricultural literature.

The Journal of the American Medical Association is more credible than the Wikipedia and Slashdot communities.

Your campus library will have access to online databases of academic journals. These databases require school subscription, so you will have to access these either on campus computers or through a VPN setup. Ask your reference librarian, and get this set up ahead of time!

Some starting points:

- Google Scholar `scholar.google.com`
- Engineering Village 2 `www.engineeringvillage2.org`
- Elsevier `www.elsevier.com`
- IEEE Transactions `ieeexplore.ieee.org`

It All Comes Down to Models

As the MCM is the Math Contest in **Modeling**, solving the problem is all about models. Developing a successful model requires identifying the central question to the problem. Attempt to distill the given problem statement into one (or several) simple questions. Keep this question in mind while telling the "story" of your solution in your paper. Focusing on the key questions may also help to identify analogous problems in different fields.

All models rely upon simplifying assumptions. Be sure that each assumption you make is explicitly identified and explained. We suggest including an entire "Assumptions" section or subsection in your paper. Try to motivate each assumption, citing work where the assumption has been made previously. When making assumptions remember that you are walking a tightrope between ignoring extraneous details and artificially changing the problem. Be sure that the key question of the problem remains unchanged.

Avoid assigning a fixed "reasonable" value to parameters that really assume a range of values: in your final analysis you should examine how your results depend on this parameter. For example, in the 2005 highway tollbooth problem, one clear parameter was the rate of incoming traffic. Understanding the behavior as this parameter varied was critical in solving the problem.

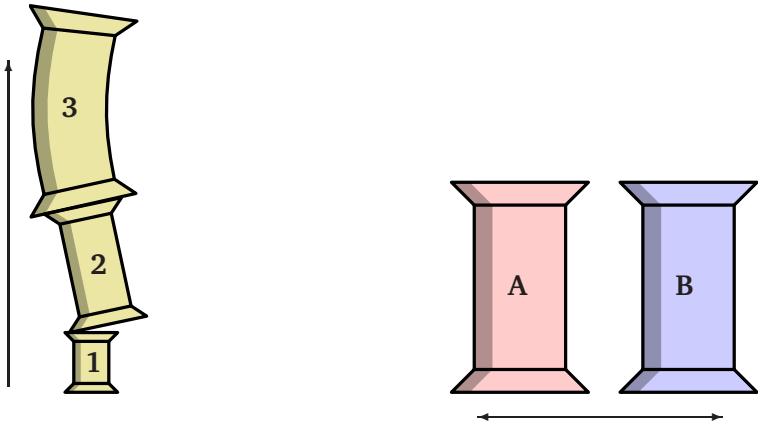
Use Multiple Models

MCM problems are usually better approached with not one, but multiple models. Compare with the literature to motivate your models and to validate your results.

A common approach to modeling is to use a sequence of increasingly refined models. Even if the first model egregiously simplifies the problem, it may provide insights to the basic behavior and inspire improved models. This progression leads to a respectable and satisfying final model, with its validity stemming from the preceding models.

At least, this is the intention. The danger with the refinement approach is that later models are too easily empirically motivated, and for this reason, ad hoc. This is especially true of models based entirely on one monolithic simulation. You must compare the results of your models with theory or against the results of other models: **check that you are not fooling yourself**. This is partly a question of good modeling, and partly a question of ethics. The more you attack your assumptions, the more rigorous your results become.

Without critical analysis and a foundational rationale, justification for the final model is shaky and top-heavy.



Refinement **Comparison**

Our preferred approach is comparing two competing models. Two disparate models are developed, possibly using different assumptions. If both models accurately represent the same problem, there ought to be concurring conclusions between the two. If these conclusions also agree with the literature, verifying similarity compels the validity of both models.

For example, typical MCM problems require a part analytic and part numerical solution. Analytic models are conclusive and general, with the flexibility to express relationships in terms of free variables, and are soundly justified by mathematical derivation. Numerical models have the advantage that they can be implemented in computer simulations, as well as providing concrete values and visualizations. Moreover, the work of developing the two models is easily split among the team members.

Evaluate Your Models With a Critical Eye

Regardless of your approach or number of models, it will be necessary to choose a quantitative formula or *criterion*[†] to rate your solution. Try to find a standard criterion from the literature. MCM problems often have industrial applications, in which case there are usually standards for measurement and acceptance. For instance, when we were told to optimize the uniformity of watering a field, we found that the irrigation industry standard is a criterion called the "coefficient of uniformity." This allowed us to see if our results were typical, and to compare them with standards from the literature.

[†]Also informally called a *metric* or *measure*, however, these two words have precise mathematical meaning.

It is often useful to consider the behavior of your model in limiting cases. If you have developed a model for multi-lane traffic, consider what it predicts when there is only one lane—is it consistent with intuition? How about data from the literature? Be sure that your model does not make absurd predictions for limiting cases.

! **Avoid dishonesty and bullshit.** In 2003, the "gamma knife" problem required teams to achieve 90% coverage of the target volume. Most teams found this to be impossible, yet the Outstanding papers acknowledged this and explained their shortcomings. They did not try to conceal the fact that they could not meet the requirement. Additionally, many teams found a useful internet source on the grassfire transform, however, many did not cite it. **Several would-be Outstanding teams were demoted for failing to cite sources.**

Write a Strong Summary

The most important page in your paper is the summary sheet (abstract). In the first round of judging, this is the only part read. **Almost half of the papers are eliminated from competition based only on their summaries.**

The summary should be written last, once all conclusions have been made. A good summary concisely states the problem, the methods for solution, and conclusions, while highlighting the merits of your approach. A summary should be more than a chronology of what you did ("A model was first developed... A simulation was implemented... It was then concluded... "). On the other extreme, the point of a summary is not to create suspense—state conclusions clearly. The summary should be brief; although a whole page is devoted to the summary, it need not and should not be completely filled.

In our approach to the summary, each team member independently writes a summary. Then as a group, the best sentences from each summary are strung together to produce another summary. This summary is elaborated, reworded, and fine-tuned, until it says everything that needs to be said as clearly and concisely as possible.

Write an Easily Skimmable Paper

In the second round of judging, judges will skim your paper. To make your paper skimmable, it must include and clearly label these sections:

- **Introduction:** Introduce the background and the problem.
- **Assumptions:** Explain all of your assumptions.
- **Conclusion:** Concisely answer the original problem.
- **Strengths & Weaknesses:** Critically assess your approach.
- **References:** Cite all references.

Additionally, make your paper more skimmable with the following compositional guidelines.

- Use sections, subsections, and sub-subsections with descriptive heading titles.
- **Use bold, etc. on key results to catch the eye.**
- Use bullet lists.
- Use figures and tables with concise captions.
- Intersperse these elements to break up long, uninterrupted lengths of text.

! **Keep figures simple: less is more.** Avoid placing a large number of figures on the same page, especially without explanation. Also, don't confuse plots and data visualizations with figures in the paper. Information-laden data visualizations are great for **your** understanding, but figures in the paper must be as simple and direct as possible. Be sure to check that all figures are legible at the scale they are printed.

Stylistic Considerations

It is common to show a developing "story" of your solution in the body of the paper. As in any technical writing, write plainly and favor shorter words over longer ones. Particularly, **it can be seen that** the **flamboyant, obdurate, and ostensibly decorous** misuse of the passive tense and excessive vocabulary tends**, thereby,** to **result in** long, awful sentences.

Among hundreds of papers, it helps if your paper has a unique, catchy title. If possible, set aside time to brainstorm paper titles. Don't use anything pretentious like *A Novel Approach to...*: Just don't. Seriously. We wouldn't mention it if it didn't keep on happening.

As part of anyone's writing education, worthwhile references are Strunk & White's *The Elements of Style* [16] and Williams' *Style: Toward Clarity and Grace* [17]. Specifically on technical writing style, see also the *Handbook of Technical Writing* by Brusaw [1].

Our basic philosophy of writing is: clarity before grammar. In this sense, we recommend Strunk & White not for its grammar rules (which are questionable), but for its own style of simplicity.

Section Summary

- Consider the paper a presentation, not a report.
- Do extensive background research.
- Use multiple models for cross-validation.
- Use organizational elements to make a skimmable paper.
- Mind your writing style, and brainstorm a catchy title.

3 A Strong Team

A common division of tasks [2] is to elect the writer, whose task is handling most of the writing, the programmer, in charge of simulations and other numerical work, and "the third," handling miscellaneous tasks and assisting the other two. All teammates should participate in the background research and model formulation, and all teammates should work together on the summary and final editing.

However, roles need not be so clearly defined. For example, our head writer also programmed, and we all contributed portions of the writing. Ideally, all three teammates can both program and write, switching between roles as necessary.

Programming

In any MCM team, at least one team member should be comfortable with a computer programming language. Prototyping languages (high-level interpreted languages like MATLAB, Python, and Java) are particularly well-suited for the contest. However, the best choice of language is one where your team can most comfortably perform the following essentials.

- **Visualize data.** Line plots, surface plots, histograms, and other means to visualize data are invaluable in understanding a problem. If your choice programming environment is graphically-limited, learn to export data to Microsoft Excel, Gnuplot, or other graphing software for visualization.
- **Numerical algorithms.** Before the contest, review numerical algorithms for fundamentals like interpolation, optimization, linear algebra, and solving differential equations. Be prepared to implement (or reuse) code for common numerical algorithms, see for example [6, 14]. Environments like MATLAB include extensive numerical routines for a variety of problems. Make use of these tools and avoid reinventing the wheel. For example, never roll your own linear algebra code. Smarter people have spent years creating LINPACK and other systems—use them.
- **Debug.** Writing code naturally involves fixing code. Know how to use the debugger tools offered by your programming environment. Use strategies like saving multiple versions, modular programming, and descriptive commenting to promote accurate code. (But don't get too carried away—your primary goal is correct results, not computer science poetry.)
- **File I/O.** Especially if your program is unstable and takes a long time to run, you need to be able to use intermediate results. A good time-saving precaution is to save progressive results to the harddrive. For example, a simulation that runs for 45 minutes could write updates to the harddrive every three minutes.

Of course, a computer-savvy team need not restrict themselves to one programming language and instead use several. Our first year, we had one person programming in MATLAB, one in Perl, and one in C with side calculations on a TI-86. However, we have found that when there is more than one programmer, sticking to one language promotes code reuse and collaboration.

Writing

Writing, as we continue to emphasize, deserves as much attention as solving the problem itself. Under the tight 4-day contest timeframe, it is vital to start writing as soon as possible, starting alongside initial research.

We recommend LaTeX as the best means for producing professional-quality scientific writing, especially as an alternative to Microsoft Word. LaTeX handles equations and mathematical symbols natively, in addition to all of the bibliographical formatting, labels and cross-referencing, and page-numbering that is ridiculously tedious in standard word-processing programs.

There are also aesthetic reasons to prefer LaTeX. Compare these compositionally equivalent samples, written in LaTeX and Word:

LaTeX

Definition 5.21 Let X be a linear space. Two norms $||\cdot||_1$ and $||\cdot||_2$ on X are called equivalent if there are constants $c > 0$ and $C > 0$ such that

$$c||x||_1 \leq ||x||_2 \leq C||x||_1, \qquad \forall x \in X.$$

Microsoft Word

Definition 5.21 Let X be a linear space. Two norms $\|\cdot\|_1$ and $\|\cdot\|_2$ on X are called *equivalent* if there are constants c > 0 and C > 0 such that

$$c\|x\|_1 \leq \|x\|_2 \leq C\|x\|_1, \quad \forall x \in X.$$

The most significant typographical difference is the spacing between lines and words. The Word sample is irregular and visually unappealing, especially around inline math, while LaTeX automatically determines line breaks and word hyphenation for aesthetically optimal spacing[†].

To get started in LaTeX, you will need the TeX typesetting system and a PS or PDF viewer. For Windows platforms, download MikTeX from `www.miktex.org` and GSview and Ghostscript from `www.cs.wisc.edu/~ghost/gsview/`. On Unix/Linux, LaTeX is already included in many distributions or available as a package. Similarly, Windows users with Cygwin (`www.cygwin.com`) can obtain LaTeX as a Cygwin package.

There are thousands of LaTeX tutorials and reference guides online; one good starting point is `www.tex.ac.uk/cgi-bin/texfaq2html`. Work through a LaTeX tutorial and write at least one practice document in LaTeX before the contest.

Regardless of whether your team uses LaTeX or not, know how to do the following:

- **Equations.** It is unavoidable that any paper will involve numerous math symbols and equations. Word users should make sure their installation includes Microsoft Equation Editor or MathType.
- **Figures.** Know how to import images and create figures. Confer with the team's programmer on transferring data visualizations into figures in the paper.
- **Section headings.** Use LaTeX's `section`, `subsection`, and `subsubsection` commands or Word's numbered headings to create consistently styled section headings. Use LaTeX's

[†]By the way, this whole article was typeset in LaTeX.

`tableofcontents` command or Word's Table of Contents feature (Insert ▶ Reference ▶ Index and Tables) to create an automatic table of contents. A good table of contents makes a paper much easier to read, and much more skimmable.

- **Citations.** Know how to write citations and bibliography entries, and be prepared with a style guide on the bibliographical formatting for various kinds of sources.

Leadership & Cooperation

While your team need not have a designated team leader, working together naturally requires group and leadership skills.

"Trust is the foundation of leadership" [11]. Trust your teammates' abilities, and respect their opinions. Give each other the freedom to work the details of their task independently. The programmer has the responsibility and authority over the details of program implementation and the head writer has the responsibility and authority over the details of the writing. While critical review of each other's work is healthy and productive, micromanagement is not.

! **Keep in frequent touch with what your teammates are doing.** If one of them—or you—is not working on a relevant task, refocus the team immediately. Avoid time-costly tasks on extraneous details and efforts that are otherwise unimportant to the paper. Nobody should ever have nothing to do—there is not enough time for that.

Conflict happens easily under high pressure and little sleep. We find that many of our disagreements are actually misunderstandings, and can be quickly resolved by open discussion. Unresolvable disagreements should be dealt with democratically among the three teammates. A poor decision is better than a late decision [11], especially on the tight MCM timescale.

Section Summary

- Be prepared to visualize data and implement numerical algorithms.
- Use LaTeX to typeset your paper.
- Trust your teammates and work cooperatively.

4 A Strong Timeline

The most demanding factor of the MCM is time. This section describes, based on our experience, a successful schedule for the contest.

We propose an intentionally front-heavy timeline, where most of the work is optimistically planned to be done by the halfway point of the contest. The main reason for this is that early work tends to be revised or discarded, mistakes and delays happen, and this timeline provides the flexibility for amendments in the later half of the contest. Furthermore, a lighter schedule in the later half means more time can be devoted to the writing.

To state our proposed timeline briefly:
On Thursday, all teammates participate in background research, formulating initial models to the problem. By Friday morning, writing begins. Late Friday to early Saturday, preliminary results are considered to revise shortcomings of the first models. By Saturday evening, the revised models yield more satisfactory results, and implications and final conclusions about the problem are drawn. The remainder of the contest is spent writing.

Sleeping

Contrary to other guides, we encourage working all-nighters on the first and third nights instead of a natural sleeping schedule. Particularly on Thursday, it is much more productive to work through the night without sleep, or to the extent that you can work without sleep. It is not necessary that all teammates are awake at the same time, just that each teammate works and sleeps productively and within their physiological limitations. In any case, don't stop early at a "natural" point.

Sleep deprivation most heavily affects creative thinking. Thought that brings together information to solve a problem is called *convergent thinking*. Thought that requires planning, originality, or unusual ideas is called *divergent thinking*. Divergent thinking is significantly impaired after one night without sleep while convergent thinking is more resilient [8]. Thus, during sleepless parts of the contest, you can productively continue tasks like programming, writing, or mathematical analysis, but brainstorming and reading should better be done when well-rested. As the contest starts with brainstorming, it is important to come into the contest fully rested.

Before the Contest

With a mere 96 hours during the contest, do what you can in advance to prepare logistically. Plan to drop all school and social activities during the MCM, and try to arrange with your instructors to reschedule classwork. On the day of the contest, arrange your team's working area, computers, food, writing templates, scratch paper, and anything else that can be done ahead of time. By the third year, we had started checking out books that *might* be useful before the contest. This paid off—we could study fluid mechanics immediately when the sprinkler problem came up.

Also prepare for the MCM mentally. Discuss old MCM problems with your team to learn how to brainstorm together. At least one teammate should prepare code and review the theory of numerical analysis fundamentals like interpolation, optimization, and solving differential equations. Ideally all teammates should learn LaTeX.

Most importantly, make sure you will be healthy and fully rested for Thursday.

Thursday: The Contest Starts

The contest officially starts at 6:00 pm[†]. At this time, the contest problems are posted at
www.comap.com/undergraduate/contests/mcm and www.mirror.comap.com/mcm.

MCM teams are given a choice between two problems. Historically, Problem A is continuous while Problem B is discrete. When choosing a problem, play to your strengths, but don't be afraid of jumping at a problem that you all find compelling. If the choice is not immediately obvious, conducting an hour or two of background research will illuminate the problem context, likely approaches, and potential difficulties.

The first task in considering a problem is to read it carefully and brainstorm possible directions. Creativity and variety are better when teammates do their initial thinking independently [3]; we recommend to delay group discussion until everyone has considered the problem on their own. Begin researching immediately, seeking online papers of previous research on your problem. This research will help identity the problem's implications, and lead you to "the right way" to think about the problem. Work aggressively through the night on research, formulating models, and programming.

[†]Our schedule is relative to GMT -7:00 US Mountain Time; adjust depending on time zone.

Friday: Formulation

Friday is perhaps the most effective day of all, spent on more research, formulating the models, significant coding, and preliminary writing. It is during this critical time that you will gain the most ground on solving the problem, so stay focused and work hard. Begin writing as soon as possible. Six hours of research and brainstorming have already passed—you should have something to say by now.

As a late morning break after a sleepless night, make a group trip to the library for journal articles and specialized resources. Try to obtain some results from the models by Friday evening. Sleep Friday night once you have preliminary results.

Saturday: Writing & Revision

Saturday should be spent on finding results and significant writing. Use your preliminary results from Friday to reconsider the models, and spend the day revising your approach and reworking results. Simplifying assumptions should be established by this point. Use the background research to support your assumptions in the paper.

Work as a team to complete the bulk of the paper. To an extent, the paper composition will guide your research. What analysis, experiments, or background research could you perform to support your claims? The writer should assign such paper subtasks to the other teammates. Don't sleep Saturday night until most of the writing is done and all of the coding is complete.

Sunday: Writing

Any new coding should stop by Sunday morning. Continue working as a team to fill in the paper. By this point, write the Strengths & Weaknesses and Conclusion sections. Once you have written out the weaknesses of your model, do what you can to fix them! Run additional tests—try justify your assumptions and explore different parameters. For the 2005 highway tollbooth problem, we started running our program with different waiting time distributions on Sunday. For the 2006 irrigation problem, we considered different "profiles" of the sprinkler.

Print a draft of the paper and read it aloud as a group. Edit and repeat. By the evening, all results must be complete and the main paper should be done. If possible, write the first draft of your summary. Sleep Sunday night.

Monday: Polish

Even if you are behind, absolutely stop adding results to the paper and work on editing. Spend most of Monday writing the summary. The summary is critical; for almost half of the contestants it is the only part read.

Write the paper appendices of additional figures, tables, and relevant code. Spend another few passes of group editing on the paper until everybody is satisfied with the writing. Don't forget to use a spell-checker at some point (Aspell, for example). When both the paper body and summary are complete, concentrate on brainstorming a good title.

The contest officially ends at 6:00 pm. Begin printing the paper with at least two hours to spare. Printers are ornery beasts that inevitably choose poor times to fail.

Section Summary

- **Before the contest:** Plan to drop everything else, prepare what you can.
- **Thursday:** Work aggressively on initial research and brainstorming.
- **Friday:** Start writing as soon as possible and get preliminary results.
- **Saturday:** Revise your approach, get the bulk of the writing done.
- **Sunday:** Stop working on the problem and work as a team on writing.
- **Monday:** Write the summary and continue editing the paper.

5 Searching for the Optimal Solution

MCM problems frequently involve optimization. In fact, some MCM problems are themselves optimization problems (both 2006 problems are optimizations—one of water uniformity, the other of profit). Even when the central focus of a problem is different, numerical optimization can still be useful. For instance, you could assume that velociraptors hunt in an "optimal" way, and use that as a starting point for describing their behavior.

We believe that many numerical analysis topics are valuable for the contest, including solving differential equations and interpolation, but that optimization deserves particular attention.

Suppose that we want to determine the "best" configuration of some system. We let x be a solution and let $\mathcal{X}$ be the set of all solutions. A solution is often formulated as an n-dimensional vector $x = (x_1, x_2, \ldots, x_n)^T$ subject to some constraints. These individual components represent parameters of your system such as number of sprinkler heads, sprinkler spacing, number of velociraptors per square kilometer, and so on.

To optimize, we need a *quantifiable* way to measure how good a solution is. That is, we need a criterion (see §2). For the sprinkler problem, this was the uniformity of watering; for the tollbooth problem, it was the traffic flow. Let $f : \mathcal{X} \to \mathbb{R}$ be this objective function. For every solution x, the objective function assigns a number $f(x)$ describing how good that solution is: the greater the value $f(x)$, the better the solution x. The best solution $x_\star$ is the solution that maximizes f,

$$f(x_\star) = \max_{x \in \mathcal{X}} f(x).$$

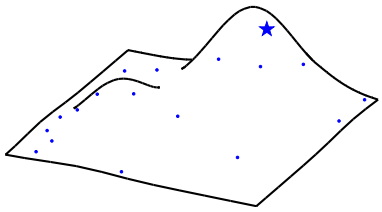
This special solution $x_\star$ is called the global optimum. However, many problems involve optimization over irregular and very large solution spaces—this may be intractable using classical techniques. **It is often essentially impossible to locate a global optimum, even in a small search space.** Instead we seek "near optimal" solutions. We describe a few algorithms for finding these solutions below:

We will need a few functions:

Function	**Description**
`rand()`	Generates a random number between 0 and 1.
`randsol()`	Generates a random solution.
`perturb(x,`δ`)`	Perturbs solution x by a random adjustment of size δ.

Random Ascent Hill Climb

```
y ← randsol()
for i = 1,..., N do
{
    x ← randsol()
    if f(x) > f(y)
        y ← x
}
```



Maximization with Random Ascent Hill Climb: try many random points. The best point found is marked with $\star$.

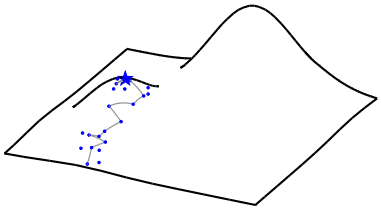
Pros: *easy to implement.*

Cons: *slow convergence.*

Steepest Ascent Hill Climb

(Choose δ small and positive)

```
y ← randsol()
for i = 1,..., N do
{
    x ← perturb(y,δ)
    if f(x) > f(y)
        y ← x
}
```



Steepest Ascent Hill Climb: advance through the solution space by randomly perturbing the best point found so far.

Pros: *faster convergence.*

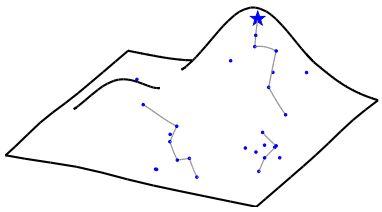
Cons: *may get trapped by local optima.*

Shotgun Hill Climb

(Choose δ small and positive)

```
z ← randsol()
for j = 1,..., M do
{
    y ← randsol()
    for i = 1,..., N do
    {
        x ← perturb(y,δ)
        if f(x) > f(y)
            y ← x
    }

    if f(y) > f(z)
        z ← y
}
```



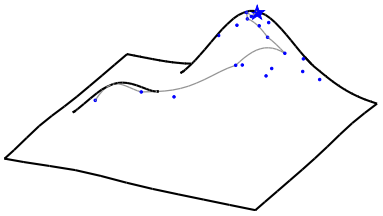
Shotgun Hill Climb: perform M Steepest Ascent searches and take the best one.

Pros: *searches more of the space.*

Cons: *need to balance N and M.*

Simulated Annealing [10]
(Choose α and β small and positive, and $p, q \geq 1$)

```
x ← randsol()
for i = 0,..., N do
{
    δ ← (1 − i/N)^p α
    η ← (1 − i/N)^q β
    y ← perturb(x, δ)
    if f(y) > f(x) − η·rand()
        x ← y
}
```



Simulated Annealing: Steepest Ascent search with decaying δ and a decaying chance to move downhill.

Pros: *can escape local optima.*

Cons: *many parameters to tune.*

These algorithms, while simple, are effective on a wide range of problems. There are possible code improvements and algorithmic variations and extensions; only the basic forms are presented. See for example [10, 15].

! **Blindly optimizing over the entire problem is a bad idea.** "Large" optimizations over many dimensions tend to be time-consuming to search and work poorly or not at all. Some precursory analysis will often reveal symmetries or other constraints that reduce the dimensionality of the solution space.

In the sprinkler problem, we started by trying a simulated annealing over the entire parameter space—number of sprinkler heads, pipe locations in three dimensions, number of pipe locations, and a few other parameters. Even if this had worked, it would have taken hours to run. After some analytical work, we restricted our optimization to just the pipe locations, which cut the calculation time down to a couple of minutes. A few minutes of thought will save hours of programming!

The algorithms described in this section belong to a class of optimization methods called "direct search methods." These are methods that only use direct evaluations of the objective function $f(x)$ and do not require evaluation of its derivatives. Consequently, direct methods are better suited for nonsmooth and high-dimensional problems than classical gradient-based search methods. Other direct methods include the Nelder-Mead simplex method [12], genetic algorithms [4, 5, 7], and particle swarm optimization [9].

On the other hand, when they do work, gradient-based methods usually have much faster convergence than direct methods. For theory and implementation, see for example [6, 13, 14].

6 Common Failures to Avoid

Our most insistent advice is marked with ! in the preceding sections. In addition, take caution against the following easy mistakes:

- **Don't charge into the contest blindly.** Someone has thought about your problem (or a related one) before. Do background research. But don't rely on only internet sources, review the literature thoroughly. Use school resources.
- **Don't rate solutions arbitrarily.** Find a metric to rate your solution, preferably a standard metric from literature.
- **Avoid arbitrary or unnecessary assumptions.** Do not egregiously over-simplify the problem or lose sight of the "key" question. For instance, the tollbooth problem was about the competition of queuing and merging. Some teams simplified the merging out of the problem!
- **Avoid modeling everything in one monolithic simulation.** Keep your thinking modular: use different models for different problems, and break a difficult problem into smaller problems. This also makes having multiple models easier.
- **Don't pick and compare anecdotal solutions.** Attempt to cover a fair amount of the solution space. Don't be afraid of brute force solutions, but realize it is unlikely that picking five "representative" solutions will sufficiently cover the search space.
- **Don't overwork yourself.** Working too long without sleep has diminishing returns. Know your limits.
- **Avoid taking large chunks of time off in the middle of the contest.** Yes, it's probably Super Bowl weekend. But a poker game or a couple of hours of TV hurts your focus—sleep is better.
- **Avoid pretentious titles.** "Conquering the Math Contest in Modeling" isn't pretentious at all.
- **Don't forget to spell-check.** Run the final paper through a spell checker before printing.
- **The summary page should not be completely filled.** Just because the space is there doesn't mean you have to use it. Brevity is a virtue.

7 Closing Remarks

The MCM is a lot of fun. It is an opportunity to think creatively, flex your math and programming skills, and get to know your teammates better. You will be surprised by how much your team can accomplish in only 96 hours.

We all greatly enjoyed our three years participating in the MCM. After our first run, we were thrilled and could hardly wait for the next contest. For all they have done, we give much thanks to Professor Dougherty and Professor Fornberg—it couldn't have happened without them.

A 2006 Questions

PROBLEM A: Positioning and Moving Sprinkler Systems for Irrigation

There are a wide variety of techniques available for irrigating a field. The technologies range from advanced drip systems to periodic flooding. One of the systems that is used on smaller ranches is the use of "hand move" irrigation systems. Lightweight aluminum pipes with sprinkler heads are put in place across fields, and they are moved by hand at periodic intervals to ensure that the whole field receives an adequate amount of water. This type of irrigation system is cheaper and easier to maintain than other systems. It is also flexible, allowing for use on a wide variety of fields and crops. The disadvantage is that it requires a great deal of time and effort to move and set up the equipment at regular intervals.

Given that this type of irrigation system is to be used, how can it be configured to minimize the amount of time required to irrigate a field that is 80 meters by 30 meters? For this task you are asked to find an irrigation algorithm that minimizes the amount of time required by a rancher to maintain the irrigation system. One pipe set is used in the field. You should determine the number of sprinklers and the spacing between sprinklers, and you should find a schedule to move the pipes, including where to move them.

A pipe set consists of a number of pipes that can be connected together in a straight line. Each pipe has a 10 cm inner diameter with rotating spray nozzles that have a 0.6 cm inner diameter. When put together the resulting pipe is 20 meters long. At the water source, the pressure is 420 Kilo-Pascals and has a flow rate of 150 liters per minute. No part of the field should receive more than 0.75 cm per hour of water, and each part of the field should receive at least 2 centimeters of water every 4 days. The water should be applied as uniformly as possible.

PROBLEM B: Wheel Chair Access at Airports

One of the frustrations with air travel is the need to fly through multiple airports, and each stop generally requires each traveler to change to a different airplane. This can be especially difficult for people who are not able to easily walk to a different flight's waiting area. One of the ways that an airline can make the transition easier is to provide a wheel chair and an escort to those people who ask for help. It is generally known well in advance which passengers require help, but it is not uncommon to receive notice when a passenger first registers at the airport. In rare instances an airline may not receive notice from a passenger until just prior to landing.

Airlines are under constant pressure to keep their costs down. Wheel chairs wear out and are expensive and require maintenance. There is also a cost for making the escorts available. Moreover, wheel chairs and their escorts must be constantly moved around the airport so that they are available to people when their flights land. In some large airports the time required to move across the airport is considerable. The wheel chairs must be stored somewhere, but space is expensive and severely limited in an airport terminal. Also, wheel chairs left in high traffic areas represent a liability risk as people try to move around them. Finally, one of the biggest costs is the cost of holding a plane if someone must wait for an escort and becomes late for their flight. The latter cost is especially troubling because it can affect the airline's average flight delay which can lead to fewer ticket sales as potential customers may choose to avoid an airline.

Epsilon Airlines has decided to ask a third party to help them obtain a detailed analysis of the issues and costs of keeping and maintaining wheel chairs and escorts available for passengers. The airline needs to find a way to schedule the movement of wheel chairs throughout each day in a cost effective way. They also need to find and define the costs for budget planning in both the short and long term.

Epsilon Airlines has asked your consultant group to put together a bid to help them solve their problem. Your bid should include an overview and analysis of the situation to help them decide if you fully understand their problem. They require a detailed description of an algorithm that you would implement which can determine where the escorts and wheel chairs should be and how they should move throughout each day. The goal is to keep the total costs as low as possible. Your bid is one of many that the airline will consider. You must make a strong case as to why your solution is the best and show that it will be able to handle a wide range of airports under a variety of circumstances.

Your bid should also include examples of how the algorithm would work for a large (at least 4 concourses), a medium (at least two concourses), and a small airport (one concourse) under high and low traffic loads. You should determine all potential costs and balance their respective weights. Finally, as populations begin to include a higher percentage of older people who have more time to travel but may require more aid, your report should include projections of potential costs and needs in the future with recommendations to meet future needs.

For more previous problems, visit

www.comap.com/undergraduate/contests/mcm/previous-contests.php.

References

[1] C. BRUSAW. *Handbook of Technical Writing.* St. Martin's Press.

[2] K. CLINE. "Kelly's Guide to the MCM." www.carroll.edu/~kcline/mcm.html

[3] GABLER, GRAY, KUCIC AND SHODHAN. "How to Prototype a Game in Under 7 Days." http://www.gamasutra.com/features/20051026/gabler_01.shtml

[4] D. BEASLEY, D. BULL, AND R. MARTIN. "An overview of genetic algorithms: Part 1, fundamentals." *University computing*, 1993, 15(2), 58-69.

[5] D. BEASLEY, D. BULL, AND R. MARTIN. "An overview of genetic algorithms: Part 2, Research topics." *University computing*, 1993, 15(4), 170-181.

[6] R. BURDEN AND J. FAIRES. *Numerical Analysis.* Brooks Cole, 2001.

[7] D. GOLDBERG. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Pub. Co., 1989.

[8] J.A. HORNE. "Sleep loss and 'divergent' thinking ability." Human Sciences Department, Loughborough University, Leicestershire, England. Dec. 1988.

[9] J. KENNEDY AND R. EBERHART. "Particle Swarm Optimization." *Proc. IEEE Int'l. Conf. on Neural Networks,* 1995.

[10] S. KIRKPATRICK, C. D. GELATT, AND M. P. VECCHI. "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[11] J. MAXWELL. *The 21 Irrefutable Laws of Leadership.* Thomas Nelson, Jan. 1998.

[12] J. A. NELDER AND R. MEAD. "A Simplex Method for Function Minimization." Comput. J. 7, 308-313, 1965.

[13] J. NOCEDAL, S. WRIGHT, S. J. WRIGHT. *Numerical Optimization.* Springer Verlag, 2006.

[14] W. PRESS, B. FLANNERY, S. TEUKOLSKY, AND W. VETTERLING. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[15] F.J. SOLIS AND R.J-B WETS. "Minimization by Random Search Techniques." *Mathematics of Operation Research* 6, pp. 19-30, 1981.

[16] W. STRUNK AND E. B. WHITE. *The Elements of Style.* Allyn & Bacon.
http://orwell.ru/library/others/style/index.htm

[17] J. WILLIAMS. *Style: Toward Clarity and Grace.* The University of Chicago Press, 1995.