MProbe: Research Plan 2001

John W. Chinneck
Systems and Computer Engineering
Carleton University
Ottawa, Ontario K1S 5B6
Canada
email: chinneck@sce.carleton.ca

August 8, 2001

Introduction

MProbe is the software implementation of a collection of tools that assist in the formulation of complex mathematical programming models. It is a utility that supports mathematical programmers in the same way that debuggers, profilers and other tools support computer programmers in standard languages such as C. The overall goal of the MProbe project is to provide helpful formulation tools for any kind of model that can be expressed in a standard mathematical programming modeling language such as AMPL, GAMS, MPL or AIMMS. Note that this includes constraint programs. More information about the project, including links to relevant papers, is available at the MProbe homepage: http://www.sce.carleton.ca/faculty/chinneck/mprobe.html.

There is now a significant need for tools of this sort. As computers have grown cheaper and more powerful, extremely large and/or complex models are being attempted with greater frequency. At the same time, novices are being introduced to mathematical programming via, for example, the solvers built into popular spreadsheet programs. In both cases, formulation assistance is needed. With very few exceptions (e.g. ANALYZE [Greenberg 1993]), the mathematical programming community has not come to grips with this pressing need.

There are two significant research challenges:

- *Inventing the algorithmic tools that can provide useful insights to modelers.* This requires original thinking given difficulties such as numerous different types of variables in a single model (real, binary, integer), and the high dimension of some functions.
- *Organizing the interface in a way that is natural and intuitive for users.* This is a difficult challenge in the face of the vast amounts of information in a large-scale industrial model.

The purpose of this document is to outline some research opportunities in this field, and to invite others to participate in this effort.

Sampling

- **Assessment of accuracy.** A basic question about the sampling methods of assessing e.g. nonlinear function shape is this: can you quantify the accuracy of the shape estimate? For example, can you indicate the probability of accurate shape assessment?

- **Sampling methodology.** More accurate estimates of model characteristics may be obtained by different sampling methods. For example, latin hypercube sampling in box enclosures may give better results, especially on functions of very high dimension. The co-ordinate directions method of hit-and-run sampling may give better results in convex sampling enclosures (see Berbee et al [1987]). Is there an advantage to increasing the sampling in important areas, e.g. near edges or near stationary points? Is there any way to determine the best way to place sample points so as to extract the maximum amount of information from the minimum number of function evaluations (see e.g. Jones et al [1998])?

- **Discrete variables.** What is the best way to sample discrete variables (binary and integer)? At present you have the option of "snapping" discrete variables by rounding the real values returned by the random number generator to the nearest discrete value. This is probably a good thing for assessing constraint effectiveness, but does not work when assessing function shape (which requires interpolation at specific real-valued points). How do you prevent the rounding from giving a non-uniform sampling? What is the best way to snap discrete variables when sampling inside a general convex enclosure rather than a box?

- **Random number generators.** A better random number generator is needed. Methods are also needed to make sure that sampling is uniform over a very large range ($\pm\ 1\times10^{20}$). Should a multi-level strategy be used? Should the range of unbounded variables be restricted even further?

- **Subdividing the sampling enclosure.** Subdividing the sampling enclosure and sampling in each of the subdivisions separately may return useful information. How should the subdivisions be decided? How should the information from the subdivisions be stitched back together?

Constraint Necessity and Redundancy

- **The Prime Analytic Centre.** The Prime Analytic Centre, may not be the best measure of the "centre" of a region. The location of the PAC can be influenced by a clustering of many necessary constraints that each have a very small surface fraction. There may be better alternatives such as e.g. the "center of gravity" idea, or biasing the effect of each necessary constraint in proportion to its surface fraction.

- **Surface fraction.** The "surface fraction" indicates the fraction of hits that a particular sampling enclosure constraint receives during hit-and-run sampling. This is a true measure of the surface fraction if the launch points of the hitting rays are uniformly distributed in the sampling enclosure and if the launch directions are uniformly distributed on the unit hypersphere. This may not be the case if we do not choose the next launch point randomly along the length of the last spanning line segment. If the rays are launched from a point approximating the Prime Analytic Center (as in the current software), then the reported surface fraction may more accurately represent the angular fraction of the enclosing surface viewed from that point. How is this information best used?

- **Identifying implied equality contraints.** A method of identifying implied equality constraints is needed. Some possible indicators of an implied equality: (i) if a spanning line segment has length zero during hit-and-run testing, (ii) two constraints have complementary results with respect to feasibility/infeasibility at all test points. How to detect implied equalities constructed from more than two constraints? There is some existing literature on this topic.

- **Redundancy relative to objective function.** Imagine a linear objective function. If you move only in the objective function gradient direction from anywhere in the feasible region, there are certain necessary constraints which are never hit because you would need to move in the reverse direction to hit them. Such constraints might be defined as "redundant relative to the objective function" in the sense that they will never be used in guiding a gradient-based method to the optimum point. This concept can be extended to nonlinear and discrete cases as well. Will this information be useful to the modeler? There may be an existing literature on this subject.

Constraint Effectiveness and Model Feasibility

- **Equality constraints.** Three figures are returned for equality constraints: (i) fraction of test points that are less than or equal to the right hand side, (ii) the fraction of test points that are equal to the right hand side (within a tolerance), and (iii) the fraction of test points that are greater than or equal to the right hand side. It may be possible to extract useful information by looking at the ratio of (i) to (iii), for example.

- **Point feasibility.** At present, MProbe considers each constraint function individually. By considering the feasibility of all constraints at a given sample point, we can gain information about the overall feasibility of the model at that point. This can also assist in shrinking the sampling zone.

Others

- **New methods of shrinking the sampling enclosure.** Testing shows that some functions pose difficulties for the existing methods of shrinking the sampling box or convex sampling enclosure (nonlinear interval analysis via sampling, finding a nucleus box, nonlinear range cutting via sampling, convex enclosure sampling, and their various combinations). There is scope for research on entirely new methods. One example: use existing software for data classification to derive hyperplanes that separate feasible from infeasible points, thereby defining a containing polytope (could also separate points below a certain objective function aspiration level from those above such a level). Shrinking the sampling box is especially important when there are unbounded variables.

- **New visualization aids.** What sort of visualizations will provide useful insights? Ideas include a radar-like sweep around a point of interest, or the ability to browse through a region of space under joystick control.

- **Design of the software interface.** A major issue is the practical usability of the software interface. Experimental work is needed to arrive at an interface that organizes the tools and resulting data in ways that are intuitive and natural for users.

- **Probing and analyzing constraint programs.** Mathematical programming and constraint programming are gradually merging, as witnessed by the migration of constraint programming structures into traditional mathematical programming languages such as AMPL. New tools for formulation assistance are needed.

- **Link to a symbolic algebra system.** There is scope for linking to a symbolic algebra system such as Maple. Many tools are possible thereafter, e.g. derivation of the Hessian, various matrix reductions, simplification of algebraic forms, etc.

- **Function approximation.** When a function is determined to be a good candidate for approximation (e.g. it is almost linear or almost convex), then methods are needed to provide the best approximation for the function (linear, piecewise linear, convex, etc.), and to return the statement of the approximation in the original modeling language.

- **Standard test set.** A set of mathematical and constraint programming models needs to be assembled as a kind of standard test set on which to exercise MProbe and related software. The models should have a range of difficulty, and should exhibit a broad range of behaviors requiring analysis.

A Note on Commercialization

The MProbe software implementation has been developed over a number of years. Because of this investment of effort, the copyright and commercialization rights will remain with me. However, I am happy to produce experimental versions of the software with interface hooks to other codes so that you can test new algorithms while avoiding the tedium of recreating an interface to the modeling languages and the other utilities available in the existing MProbe. Useful new routines can certainly be packaged for sale as add-ins to MProbe.

References

Berbee, Boender, Rinooy Kan, Scheffer, Smith, Telgen (1987), *Hit-and-run algorithms for the identification of nonredundant linear inequalities*, Math Programming, vol 37, no 2.

D.R. Jones, M. Schonlau, W.J. Welch (1998), *Efficient Global Optimization of Expensive Black-Box Functions*, Journal of Global Optimization, vol 13, pp. 455-492.

H.J. Greenberg (1993), **A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions: A User's Guide for ANALYZE**, Kluwer Academic Publishers, Norwell Massachusetts.