Nonlinear Optimization in Mathematica with MathOptimizer Professional

János D. Pintér[1] and Frank J. Kampas[2]

1. Pintér Consulting Services, Inc., 129 Glenforest Drive, Halifax, NS B3M 1J2, Canada, jdpinter@hfx.eastlink.ca, www.pinterconsulting.com
2. WAM Systems, Inc., 600 West Germantown Pike, Suite 230, Plymouth Meeting, PA 19462, USA, fkampas@wamsystems.com, www.wamsystems.com

Abstract

The MathOptimizer Professional software package combines *Mathematica*'s modeling capabilities with an external solver suite for general nonlinear – global and local – optimization. Optimization models formulated in *Mathematica* are directly transferred along to the solver engine; the results are seamlessly returned to the calling *Mathematica* document. This approach combines the advantages of a high-level modeling system with the Lipschitz Global Optimizer (LGO) algorithm system that has been in use for over a decade.

We summarize the key global optimization concepts, then provide a brief review of LGO and an introduction to MathOptimizer Professional (MOP). MOP functionality is illustrated by solving small-scale, but non-trivial numerical examples, including several circle packing models.

1. *Mathematica* and its application perspectives in operations research

Mathematica is one of the most ambitious and sophisticated software products available today. Its capabilities and a broad range of applications are documented in *The Mathematica Book* [1], as well as in hundreds of books and many thousands of articles and presentations. Visit the website of Wolfram Research and the *Mathematica* Information Center [2] for extensive information and links.

Operations Research (OR) and Management Science (MS) apply advanced analytical modeling concepts and optimization methods to help make better decisions. The general subjects of OR/MS modeling and optimization are described in references [3-10]. The INFORMS website [11] and numerous other sites also provide topical information, with further links and documents.

Mathematica can be put to good use also in the context of OR/MS. Among *Mathematica*'s many capabilities that are particularly useful are the following: support for advanced numerical and symbolic calculations, rapid prototyping, concise code development, model scalability, external link options (to application packages, to other software products, and to the Internet), and the possibility of integrated documentation, calculation, visualization, and so on within 'live' *Mathematica* notebooks that are fully portable across hardware platforms and operating systems. These features can be used effectively in the various stages of developing OR/MS applications. Model formulation, data analysis, solution strategy and algorithm development, numerical solution and related analysis, and project documentation can be put together from the beginning in the same *Mathematica* document (if the developer

Such notebooks can also be directly converted to TEX, html, xml, ps, and pdf file formats.

Let us note here that, although optimization modeling and solver environments – from Excel spreadsheets, through algebraic modeling languages (AIMMS, AMPL, GAMS, LINGO, MPL), to integrated scientific-technical computing systems (*Mathematica*, Maple, MATLAB) – may have a slower program execution speed when compared to a compiled 'pure number crunching' solver system, the overall development time can be massively reduced by using such systems to develop optimization applications. It is instructive to recall in this context the debate that surrounded the early development of programming languages (ALGOL, BASIC, FORTRAN, PASCAL, and so on), as opposed to machine-level assembly programming.

Within the broad category of OR/MS modeling and optimization, we see particularly strong application potentials for *Mathematica* in nonlinear systems analysis and optimization. Nonlinear objects, formations and processes are ubiquitous, and are studied in the natural sciences, engineering, economics and finances. Nonlinear system models may not be easy to cast into a pre-defined, 'boxed' formulation: instead, problem-specific modeling and code development are often needed, and using *Mathematica* as the model development platform can be very useful. Some interesting discussions of nonlinear system models and their applications are given in references [12-20]. The more strictly topical global optimization literature discusses nonlinear models and their applications which require a global solution approach [21-31].

2. The global optimization challenge

We shall consider the continuous global optimization (CGO) model:

$$\min f(x) \text{ subject to } x \in D \subset \mathbb{R}^n \quad D := \{x : xl \le x \le xu;\ g(x) \le 0\}. \tag{1}$$

Here $x \in \mathbb{R}^n$ is the vector of decision variables ($\mathbb{R}^n$ is the Euclidean real n-dimensional space); $f : \mathbb{R}^n \to \mathbb{R}^1$ is the objective function. The set of feasible solutions $D \subset \mathbb{R}^n$ is defined by finite (component-wise) lower and upper bounds xl and xu, and by a finite collection of constraint functions $g : \mathbb{R}^n \to \mathbb{R}^m$; the vector-inequality $g(x) \le 0$ is interpreted component-wise (i.e., $g_j(x) \le 0$, $j = 1, \ldots, m$).

Let us remark that the relations $g(x) \le 0$ directly cover the formally more general case $g_j(x) \sim 0$, where $\sim$ symbolizes any one of the relational operators $=$, $\le$, and $\ge$. In addition, the model (1) also covers formally the more general mixed integer case. To see this, it is sufficient to represent first each finitely bounded integer variable y in binary form $y = \sum_{i=0}^{imax} 2^i y_i$. Here $imax = imax(y) \le \lceil \log_2 y \rceil$ (that is, $imax(y)$ is found by rounding $\log_2 y$ to the next majorant integer). Then one can replace each of the binary variables y_i by the continuous relaxation $0 \le y_i \le 1$ and the added non-convex constraint $y_i(1 - y_i) \le 0$. This note immediately implies the inherent computational complexity and resulting general numerical challenge in CGO, since all combinatorial models are formally subsumed by the model (1).

If the model functions f and g (the latter component-wise) are all convex, then by classical optimization theory local optimization approaches would suffice to find the globally optimal solution(s) in (1). For example, Newton's method for finding the unconstrained minimum of a positive definite quadratic function, and its extensions to more general constrained optimization (by Lagrange, Cournot, Karush, Kuhn, Tucker, and others) are directly applicable in the convex case. Unfortunately, model convexity may not hold, or provably does not hold, in many practically relevant cases.

Let us emphasize that without further structural assumptions the general CGO model can lead to very difficult numerical problems (even in low dimensions such as $n = 1$ or 2). For instance, D could be disconnected, and even some of its component subsets could be non-convex; furthermore, f could be highly multiextremal. Consequently, model (1) may have a typically unknown number of global and local optima. There is no general algebraic characterization of global optimality, without proper consideration to the entire CGO model structure. By contrast, in 'traditional' nonlinear programming, most exact algorithms aim at finding solutions to the Karush-Kuhn-Tucker system of (necessary, local) optimality conditions: the corresponding system of equations and inequalities in the present framework becomes another GO problem, often at least as complex as the original model. Furthermore, increasing dimensionality (n, m) could lead to exponentially increasing model complexity: with respect to this point see the related discussion by Neumaier [32].

Let us also note that the mere continuity assumptions (related to f and g) do not support the calculation of deterministically guaranteed lower bounds for the optimum value. To establish a valid bounding procedure, one can assume (or postulate) Lipschitz-continuous model structure. For example, if f is Lipschitz-continuous in $[xl, xu]$, i.e. for an arbitrary pair of points x^1 and x^2 we have $| f(x^1) - f(x^2) | \le L \| x^1 - x^2 \|$, then a single point x in $[xl, xu]$ and the corresponding function value $f(x)$ will allow the computation of a lower bound for f over the entire box $[xl, xu]$. (Here we assume that $L = L([xl, xu], f) \ge 0$ is a known overestimate of the, typically unknown, smallest possible Lipschitz-constant for f over $[xl, xu]$.)

Of course, we do not claim that all CGO model instances are extremely difficult. However, in numerous practically relevant cases we may not know *a priori*, or even after a possibly limited inspection and sampling procedure, how difficult the actual model is: one can think of nonlinear models defined by integrals, differential equations, special functions, deterministic or stochastic simulation procedures, and so on. In fact, many practically motivated models indeed have a difficult multiextremal structure [17, 19, 21-31]. Therefore, global numerical solution procedures have to cope with a broad range of CGO models.

The CGO model class includes a number of well-structured, specific cases (such as e.g., concave minimization under convex constraints), as well as far more general problems, e.g. differential convex, Lipschitz-continuous or merely continuous. Therefore one can expect that the corresponding 'most suitable' solution approaches will also vary to a considerable extent. On one hand, a general optimization strategy should work for broad model classes, although its efficiency might be lower for more special instances. On the other hand, highly tailored algorithms may not work for problem-classes outside of their intended scope. The next section discusses the LGO solver suite that has been developed with the objective of handling in principle 'all' models from the CGO model-class.

3. The LGO solver system

In theory, we want to find all global solutions to a CGO model instance, tacitly assuming (or postulating) that the solution set X^* is at most countable. The practical objective of global optimization is to find suitable numerical approximations of X^*, and of the corresponding optimum value $f^* = f(x^*)$ for $x^* \in X^*$.

There are several logical ways to classify global optimization strategies. Natural dividing lines lie between exact and heuristic methods, as well between deterministic and stochastic algorithms. Here we will comment only on exact methods: for related expositions see references [26, 30, 31, 33, 34].

Exact deterministic algorithms, at least in theory, have a rigorous guarantee for finding at least one

global solution, or even all points of X^*. However, the associated computational burden rapidly may become excessive, especially for higher-dimensional models, and/or for more complicated model functions. Most CGO, specifically including also combinatorial, models are known to be NP-hard (that is, their solution can be expected to require an exhaustive search in the worst case). Therefore even the fast increase of computational power will not resolve their fundamental numerical intractability. For this reason, in higher dimensions and without special model structure, there is arguably more practical hope in applying exact stochastic algorithms, or other methods which have (also) a stochastic global search component. This observation has played a key role in the gradual development of the LGO solver system that is aimed at the numerical solution of a broad range of instances of the general model (1).

LGO, originally referring (only) to the Lipschitz Global Optimizer solver component, has been developed and maintained since 1990, and it is discussed in more detail elsewhere. For theoretical background see reference [26]; references [27, 28, 35] discuss more recent implementations. The software review [36] discusses one of these, the MS Windows 'look and feel' LGO development environment.

Here we provide only a concise summary of the key concepts and solver components. LGO integrates a suite of global and local nonlinear optimization algorithms. The current LGO implementation includes the following solver modules:

- Branch-and-bound global search method (BB)
- Global adaptive random search (GARS) (single-start)
- Multi-start based global random search (MS)
- Local search by the generalized reduced gradient method (LS).

All component solvers are implemented derivative-free: this is of particular relevance with respect to applications, in which higher order (gradient, Hessian,…) information is impossible, difficult, or costly to obtain. This feature makes LGO applicable to a broad range of 'black box' (complicated, possibly confidential, and so on) models, as long as they are defined by continuous model functions over finite n-dimensional intervals.

The global search methodology and convergence results underpinning BB, GARS, and MS are discussed in [26, 37], with extensive further pointers to relevant literature. All three global scope algorithms are automatically followed by the local search algorithm. The generalized reduced gradient method is discussed in numerous textbooks with an emphasis on local optimization [8]. Therefore only a concise review of the proprietary LGO solvers is included below, with notes related to their implementation.

The BB solver component implements a theoretically convergent global optimization approach, for Lipschitz-continuous functions f and g. BB combines set partition steps with deterministic and randomized sampling within subsets. The latter strategy supports a statistical bounding procedure that, however, is not rigorous in a deterministic sense (since the Lipschitz information is not known in most cases). The BB solver module will typically generate a close approximation of the global optimizer point(s), before LGO switches over to local search. Note that LGO runtimes can be expected to grow in higher-dimensional and more difficult models, if we want to find a close approximation of the global solution by BB alone. (Let us remark that similar comment applies also with respect to all other theoretically exact, implementable global search strategies.)

Pure random search is a simple 'folklore' approach that converges to the global solution (set) with

probability one, under mild analytical conditions: this includes models with merely continuous functions f and g, without the guaranteed Lipschitz-continuity assumption. The GARS solver mode is based on pure random search, but it adaptively attempts to focus the global search effort on the region which, on the basis of the actual sample results, is estimated to contain the global solution point (or, in general, one of these points). Similarly to BB, this method generates an initial solution for subsequent local search.

Multi-start based global search applies a similar search strategy to single-start; however, the total sampling effort is distributed among several global searches. Each of these leads to a 'promising' starting point for subsequent local search. This strategy can be recommended in presence of a possibly large number of competitive local optima. Typically, MS requires the most computational effort (due to its multiple local searches); however, in complicated models, it often finds the best numerical solution.

Note that all three global scope methods search over the entire range $xl \le x \le xu$, while jointly considering the objective function f and the constraints g. This is done by introducing internally an aggregated exact l_2-penalty function defined as:

$$f(x) + \sum_{j \in E} \left| g_j(x) \right|^2 + \sum_{j \in I} \max(g_j(x), 0)^2 \qquad (2)$$

here the index sets E and I of the summation signs denote the sets of equality and inequality constraints. The numerical viability of this approach is based on the tacit assumption that the model is at least 'acceptably' scaled. A constraint penalty parameter $p > 0$ can be adjusted in the LGO solver options file: this value is used to multiply the constraint functions, to give more or less emphasis to the constraints.

Ideally, and also in many practical cases, the three global search methods outlined above will give the same answer, except small numerical differences due to rounding errors. In practice, when trying to solve complicated problems with a necessarily limited computational effort, the LGO user may wish to try all three global search options, and then to compare the results obtained.

The local search mode, following the standard convex nonlinear optimization paradigm, starts from the given (user-supplied, or global search based) initial solution, and then performs a local search. As mentioned above, this search phase is currently based on the generalized reduced gradient algorithm. A key advantage of the GRG method is that once it attains feasibility, it will maintain it (save numerical errors and other unforeseen problems). Therefore its use can also be recommended, as a stand-alone dense nonlinear solver. The application of the local search mode typically results in an improved solution that is at least feasible and locally optimal.

The solver suite approach implemented in LGO supports the numerical solution of both convex and non-convex models under mild (continuity) assumptions. LGO even works in cases, when some of the model functions are discontinuous at certain points over the box region [xl, xu]. The MathOptimizer Professional User Guide [38] provides further details, including modeling and solver tuning tips.

Without going into details unnecessary in the present context, let us remark that LGO is currently available for numerous professional C and Fortran compiler platforms, and as a solver engine with links to Excel, GAMS, MPL, Maple, *Mathematica* and TOMLAB (MATLAB). The MPL/LGO solver, in a demo version, now accompanies the well-received textbook [9].

4. MathOptimizer Professional

MathOptimizer Professional (MOP) is a third party software package [38] for global and local nonlinear optimization. MOP can be used with recent *Mathematica* versions (4 and higher): it is currently set up for Windows platforms, but other implementations can also be made available. In our illustrative tests discussed here, we use *Mathematica* 5.0 and 5.1, and P4 machines running under Windows XP Professional.

MathOptimizer Professional combines *Mathematica*'s modeling capabilities with the LGO solver suite: optimization models formulated in *Mathematica* are directly sent (via *MathLink*) to LGO; the optimization results are returned to the calling *Mathematica* notebook.

The overall functionality of MOP can be summarized by the following steps:

- formulation of constrained optimization model, in a corresponding *Mathematica* notebook,
- translation of the *Mathematica* model into C or Fortran code,
- compilation of the model code into a Dynamic Link Library (DLL); this step obviously requires a suitable compiler,
- call to the external LGO engine, an executable program that is linked together with the model DLL,
- model solution and report generation by LGO, and
- LGO report display within the calling *Mathematica* notebook.

Note that the steps above are fully automatic, except (of course) the model formulation in *Mathematica*. The approach outlined supports (only) the solution of models defined by *Mathematica* functions that can be directly converted into C and Fortran program code. However, this still allows the handling of a broad range of continuous nonlinear optimization models (including, of course, all such models that could be described in C and Fortran). Let us also remark that a 'side-benefit' of using MOP is its automatic C or Fortran code generation feature: this can be put to good use e.g. in generating test model libraries, or in application development.

We have tested MathOptimizer Professional in conjunction with various C and Fortran compilers, including Borland C/C++ version 5, Lahey Fortran 90 and 95; Microsoft Visual C/C++ version 6.0, Salford Fortran FTN77 and FTN95. Note that except some (internal) naming convention differences all MOP functionality is compiler-independent; versions for other compiler platforms can easily be made available upon request.

MathOptimizer Professional can be launched by the following *Mathematica* command:

```
Needs["MathOptimizerPro`callLGO`"];
```

Upon execution of the above statement, a command window opens for the *MathLink* connection to a system call option. (One could trace in this window the compiler and linker messages; however, in case of errors corresponding messages appear also in the *Mathematica* environment.)

The basic functionality of **callLGO** can be queried by the following *Mathematica* statement: see the auto-generated reply immediately below.

```
? callLGO
```

```
callLGO[obj_, cons_List, varswithbounds_List, opts___]:
obj is the objective function,
cons is a list of the constraints,
varswithbounds are the variables and their
  bounds in the format {{variable, lower bound,
  initial value for local search, upper bound}...}
  or {variable, lower bound, upper bound}...}.
Function return is the value of the objective
  function, a list of rules giving the solution,
  and the maximum constraint violation.
See Options[callLGO] for the options and also see
  the usage statements of the various options for
  their possible values. For example, enter ?Method
  for the possible settings of the Method option.
```

```
Options[callLGO]
```

```
{ShowSummary -> False, Method -> MSLS,
 MaxEvals -> ProblemDetermined,
 MaxNoImprov -> ProblemDetermined,
 PenaltyMult -> 1, ModelName -> "LGO Model",
 DllCompiler -> VC, ShowLGOInputs -> False,
 LGORandomSeed -> 0, TimeLimit -> 300,
 TOBJFGL -> -1000000, TOBJFL -> -1000000,
 MFPI -> 1.*^-6, CVT -> 1.*^-6, KTT -> 1.*^-6}
```

The box below briefly explains the current **callLGO** options. All options can be changed by the user, as per given specifications.

option name	*default value*	*Description*
`ShowSummary`	`False`	Do not display (or display) `LGO` report file (`LGO.SUM`).
`Method`	`MSLS`	Multistart global search (`MS`) followed by local search (`LS`). Alternative choices are `BBLS`, `GARSLS`, `LS`.
`MaxEvals`	`ProblemDetermined`	Allocated global search effort, set by default to 1000 ($n + m$) (global search phase stopping criterion).
`MaxNoImprov`	`ProblemDetermined`	Allocated global search effort w/o improvement, set by default to 200 ($n + m$) (global search phase stopping criterion).
`PenaltyMul`	1	Penalty multiplier.
`ModelName`	`LGO Model`	Model−dependent (default) name.
`DllCompiler`	`MSVC`	Supported compilers : Borland/Lahey/Microsoft/Salford.

callLGO options (continued overpage)

option name	*default value*	*Description*
`ShowLGOInputs`	`False`	Do not display (or display) the generated `LGO` input files.
`LGORandomSeed`	0	Set to generate randomized search points.
`TimeLimit`	300	Maximal allowed runtime (in seconds) (global search phase stopping criterion).
`TOBJFGL`	−1000000	Target objective function value in global search phase (global search phase stopping criterion).
`TOBJFL`	−1000000	Target objective function value in local search phase (stopping criterion).
`MFPI`	10^{-6}	Merit function precision improvement tolerance (local search phase stopping criterion).
`CVT`	10^{-6}	Accepted constraint violation tolerance (local search phase stopping criterion).
`KTT`	10^{-6}	Kuhn−Tucker condition tolerance (local search phase stopping criterion).

callLGO options

5. Circle packing problems

5.1 Introductory notes

A circle packing is a non-overlapping arrangement of circles inside a given set in $\mathbb{R}^2$. This general framework specifically includes the frequently studied case when all circles have the same (*a priori* unknown, optimized) radius, and the circles are packed into the unit square or the unit circle. In this section, illustrative numerical results are presented for these two special cases, as well as for the more general case of packing non-uniform (in principle, arbitrary) size circles into an optimized circle.

Let us remark first that, in general, packings lead to difficult GO problems that can not be handled by classical numerical approaches. Even the much simpler cases of packing identical size circles into a square or into a circle are solved only for a few instances (up to a few tens of circles), to analytical or proven numerical optimality. This is in spite of the significant effort spent on variants of this problem by mathematical institutes and enthusiastic hobbyists for decades. From the massive packing literature, we refer here only to Melissen's thesis [39] that provides extensive further pointers, and to the topical website of Specht [40] that cites best known results and related references.

Let us also emphasize that we will not exploit any prior structural considerations and heuristic initial arrangements that could help (significantly) in the optimization, since we wish to illustrate the general solver capabilities of MathOptimizer Professional. Although the case of packing uniform size circles is often studied, the optimization approach advocated here can be directly brought to the more difficult case of non-uniform size circle packings, as well as to other similar problems. To our best knowledge, the case of non-uniform circle packings, as introduced here, has not been investigated by other researchers in optimization.

A Google web search with keywords 'packing problems' leads to hundreds of thousands of hits: many of these hits indicate also the practical relevance of the subject. Uniform circle packings are closely related to various models arising in numerical analysis, experimental design, computational physics and chemistry, and in medical studies. The non-uniform circle packing model, and more general object packings, have practical applications e.g. in ergonomic design, dashboard design [41] or in actual (physical) packings [42].

We shall consider k circles with arbitrary radii $r(i) > 0$, for $i = 1, \ldots, k$. Equal size circles can be considered as a special case of the general modeling framework presented below. We shall assume that the centers of the circles have coordinates $x(i)$ and $y(i)$ for $i = 1, \ldots, k$: these are decision variables to be determined. In addition, in the cases with identical circles, we will optimize their radius; in the general circle packing case, we will optimize the radius of the circumscribing circle.

5.2 Non-overlapping circle arrangements

The condition that circles i and j must not overlap is equivalent to requiring that the sum of their radii can not be greater than the Euclidean distance between their centers:

$$r(i) + r(j) \le \sqrt{(x(i) - x(j))^2 + (y(i) - y(j))^2} \quad \text{for } i, j = 1, \ldots, k \text{ and } i < j \tag{3}$$

Next we shall summarize the objective function and constraints for the three types of problems considered here. In addition to the 'no-overlap' constraints formulated above, a second set of constraints postulate that the circles are inside the enclosing circle or square. The latter constraints are also derived from simple geometrical considerations.

5.3 Packing equal size circles in the unit square

We shall consider first the case of packing identical circles in the unit square that is, by assumption, centered around the origin. Then the optimization model is formulated as follows:

$$\max r \tag{4}$$

$$\left| x(i) \right| + r \le \frac{1}{2} \qquad \text{for } i = 1, \ldots, k \tag{5}$$

$$\left| y(i) \right| + r \le \frac{1}{2} \qquad \text{for } i = 1, \ldots, k \tag{6}$$

$$2r \le \sqrt{(x(i) - x(j))^2 + (y(i) - y(j))^2} \quad \text{for } i, j = 1, \ldots, k \text{ and } i < j \tag{7}$$

Observe that we have $2k$ nonlinear convex constraints, and $k(k-1)/2$ non-convex constraints (since the norm function itself is convex). A *Mathematica* function which implements this model, returning the objective function, constraints, and variables with bounds is as follows:

```
In[1]:= circumscribecircles[k_Integer] :=
         Module[
          (* local variables in the module *)
          {cons, cons1, cons2, cons3, vars, lb, ub, soln, r},
```

```
(* the set of no-overlap constraints  *)
cons1 =
 Flatten [Table[ Sqrt[(x[i] - x[j])^2 + (y[i] - y[j])^2] ≥ 2 r,
   {i, k - 1}, {j, i + 1, k}]];

(* the constraint set which keeps the circles inside -1/2 ≤
  x < 1/2 *)
cons2 = Table[r + Abs[x[i]] ≤ 1/2, {i, k}];
(* the constraint set which keeps the circles inside -1/2 ≤
  y < 1/2 *)
cons3 = Table[r + Abs[y[i]] ≤ 1/2, {i, k}];
(* the variables are the circle center coordinates
  and the radius of the identical circles *)
vars = Join[Flatten[Table[{x[i], y[i]}, {i, k}]], {r}];
(* join all the constraints *)
cons = Join[cons1, cons2, cons3];
(* lower and upper bounds on the variables *)
lb = Join[Table[-1, {2 * k}], {0}];
ub = Join[Table[1, {2 * k}], {1. / Sqrt[π k]}];
(* the objective function (negative since we want
    to maximize r, constraints and variables *)
{-r, cons, Transpose[{vars, lb, ub}]}
]
```

5.4 Packing equal size circles in the unit circle

Next, we consider the case of packing identical circles in the unit circle with its center at the origin. Then the optimization model is:

$$\max r \tag{8}$$

$$\sqrt{x(i)^2 + y(i)^2} + r \le 1 \text{ for } i = 1, \ldots, k \tag{9}$$

$$2r \le \sqrt{(x(i) - x(j))^2 + (y(i) - y(j))^2} \quad \text{for } i, j = 1, \ldots, k \text{ and } i < j \tag{10}$$

Consequently, now we have k convex nonlinear constraints, in addition to the $k(k-1)/2$ non-convex constraints. The *Mathematica* model is very similar to the model for the unit square.

```
In[2]:= circumscribecircles[k_Integer] :=
  Module[
   {cons, cons1, cons2, vars, lb, ub, soln, r},
   cons1 =
```

```
 Flatten [Table[ Sqrt[(x[i] - x[j])^2 + (y[i] - y[j])^2] ≥ 2 r,
    {i, k - 1}, {j, i + 1, k}]];
 (* the constraint  which keeps the circles
   inside  the circumscribing  circle  *)
 cons2 = Table[r + Sqrt[x[i]^2 + y[i]^2] ≤ 1, {i, k}];

 vars = Join[Flatten[Table[{x[i], y[i]}, {i, k}]], {r}];
 cons = Join[cons1, cons2];
 lb = Join[Table[-1, {2 * k}], {0}];
 ub = Join[Table[1, {2 * k}], {1. / Sqrt[k]}];
 {-r, cons, Transpose[{vars, lb, ub}]}
]
```

5.5 Packing circles of various size into the smallest circumscribing circle

In this case, we have to find the circle of optimized radius, with its center at the origin, that contains all the given circles in a non-overlapping arrangement. The corresponding optimization model is:

$$\min r \tag{11}$$

$$\sqrt{x(i)^2 + y(i)^2} + r(i) \le r \qquad \text{for } i = 1, \ldots, k \tag{12}$$

$$r(i) + r(j) \le \sqrt{(x(i) - x(j))^2 + (y(i) - y(j))^2} \quad \text{for } i, j = 1, \ldots, k \text{ and } i < j \tag{13}$$

Similarly to the model above, we have k convex (nonlinear) constraints and $k(k-1)/2$ non-convex constraints.

The above model formulations imply that the complexity of all three packing models increases rapidly, as k increases. The *Mathematica* function is more complex than the previous ones due to the different sizes of the circles.

```
In[3]:= circumscribecircles[k_Integer] :=
  Module[
   {cons, cons1, cons2, vars, lb, ub, soln, rcirc},
   (* generate  the radii  of the k circles  *)
   Do[r[i] = i^(-1/2), {i, k}];

   cons1 = Flatten [
     Table[r[i] + r[j] ≤ Sqrt[(x[i] - x[j])^2 + (y[i] - y[j])^2],
      {i, k - 1}, {j, i + 1, k}]];
   (* constrain  the n circles  to be inside
     the circumscribing  circle  *)
   cons2 = Table[rcirc - r[i] ≥ Sqrt[x[i]^2 + y[i]^2], {i, k}];
```

```
 vars =
  Join[Flatten[Table[{x[i], y[i]}, {i, k}]], {rcirc}];
 cons = Join[cons1, cons2];
 lb = Join[Table[-2.5, {2 * k}], {1}];
 ub = Join[Table[2.5, {2 * k}], {3}];
 {rcirc, cons, Transpose[{vars, lb, ub}]}
]
```

5.6 Illustrative numerical results

In this section we present some numerical results for the three packing models defined in the previous section. These results have been calculated using MOP in conjunction with the Salford FORTRAN FTN 95 compiler. The computer used has a 3.2 GHz Pentium Pro processor and it is running under Windows XP. *Mathematica* graphics primitives were used in constructing the results shown.

In the case of packing 20 identical size (optimized) circles in the unit square we have obtained the radius $r \sim 0.1113823475$. This value agrees to at least 10^{-10} absolute precision with the best known value 0.111382347512 posted at www.packomania.com. Figure 1 displays the arrangement found. As per our related notes, this 20 circle model has $2k + 1 = 41$ decision variables, $2k = 40$ nonlinear convex constraints, and $k(k-1)/2 = 190$ non-convex constraints. The corresponding MOP runtime is approximately 18 seconds.

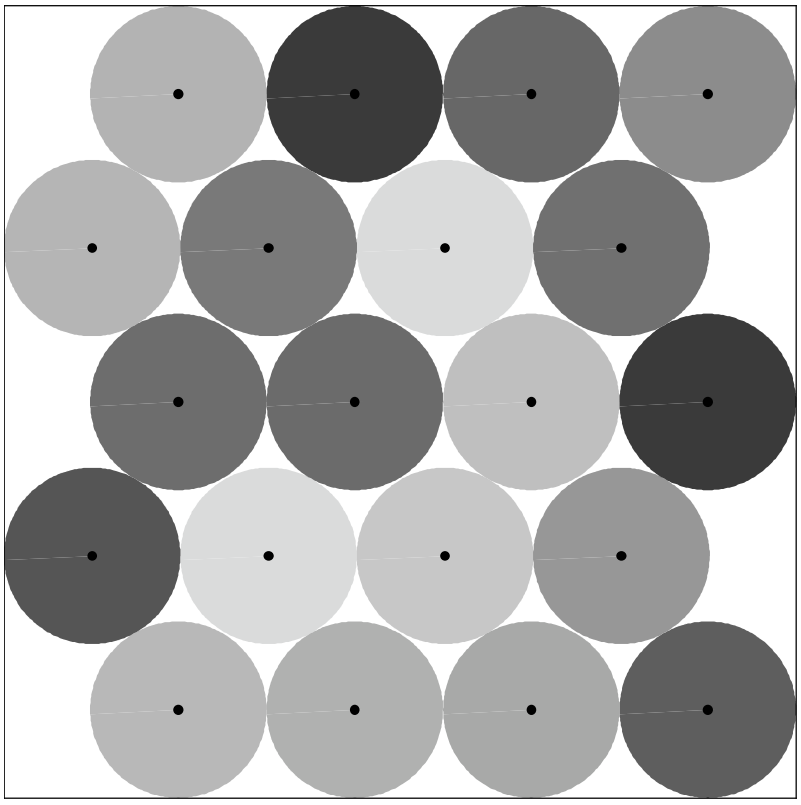

Fig. 1 Optimized configuration of 20 identical size circles in the unit square.

In the next example, for packing 20 identical size circles in the unit circle we have obtained the (optimized) radius $r \sim 0.1952240110$. Again, this value agrees to at least 10^{-10} absolute precision with the best known value 0.195 224 011 018 748 878 291 305 694 833 posted at www.packomania.com. Figure 2 displays the arrangement found. This 20 circle model has $2k + 1 = 41$ decision variables, $k = 20$ nonlinear convex constraints, and $k(k-1)/2 = 190$ non-convex constraints. The corresponding MOP runtime is approximately 43 seconds.

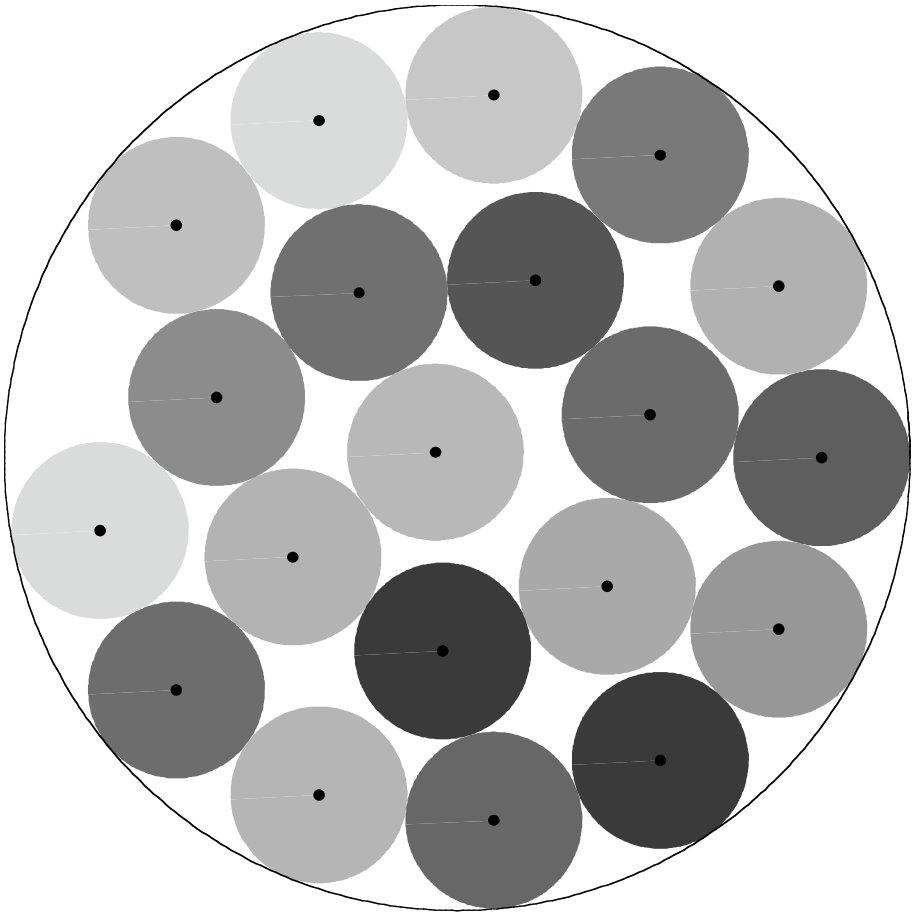

Fig. 2 Optimized configuration of 20 identical size circles in the unit circle.

In the last numerical example shown here, we have been packing circles of size $i^{-1/2}$ for $i = 1, \ldots, k$ into a circumscribing circle that is centered at the origin, and minimized its radius. Note that the total area of the packed circles is slowly divergent as k goes to infinity (since their total area is $\pi \sum_{i=1}^{k} 1/i$): therefore the optimized radius also goes to infinity as k grows. Figure 3 shows the configuration found: the radius of the circumscribing circle is $r \sim 2.1246798149$. The MOP runtime is 47 seconds: this is only about 10 percent higher than for solving the special case of uniform size circles (for the same number of circles).

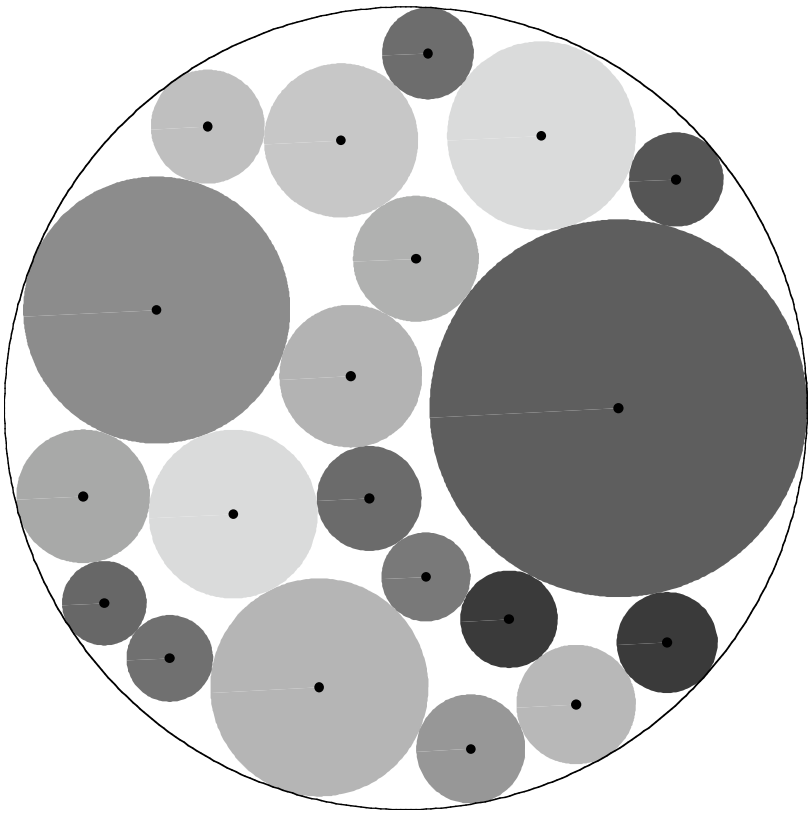

Fig. 3 Optimized configuration of 20 non-uniform size circles in a circle.

Let us point out again that structural considerations and heuristic initial arrangements have not been used to generate these results. This fact illustrates the core (default) numerical capabilities of MathOptimizer Professional, at least for the examples presented in this paper. So far, we solved packing models for up to $k = 40$ circles. The longest runs take 3 to 4 hours on P4 processor based computers [43]. If we wished to solve larger model instances, then some more advanced modeling (and perhaps also heuristic) considerations should also be exploited, to speed up the solution procedure.

From an optimization standpoint, packing different size circles into a circumscribing circle is a much more complex problem than packing equal size circles since there are many local minima close in value to the global minimum. Many of these local minima can be regarded as arising from interchanging two circles in the global solution and then suitably 'readjusting' the other circles. Such interchanges have no effect on the solution when packing equal size circles. Therefore packing unequal size circles can be used to test global optimizers for the class of problems that have a large number of local minima close in value to the global minimum: such models occur frequently e.g. in computational physics and chemistry.

The results of such a test are shown in Figures 4 and 5.The time required and the objective function values obtained for 10 sequentially performed optimizations (using different random seeds) are shown for packing 3 to 10 circles, for both MathOptimizer Professional and the built-in *Mathematica* function **NMinimize**. The **Multistart** method was used for MathOptimizer Professional and the **Differential Evolution** method was used for **NMinimize** as these methods give the best results for this class of problems for the two optimizers. In both figures, MathOptimizer Professional performance is shown by black (darker) dots, and **NMinimize** performance is shown by red (lighter) dots.

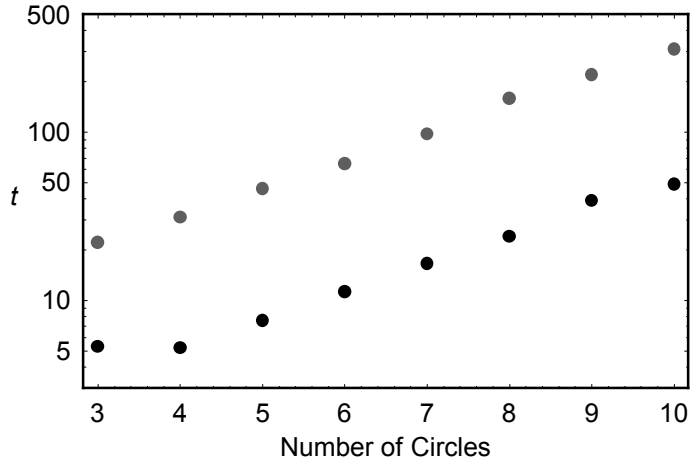

Fig. 4 Timing comparison (seconds) between MathOptimizer Professional and **NMinimize** for unequal circle packing. The results are based on 10 sequentially performed optimizations, using different random seeds.

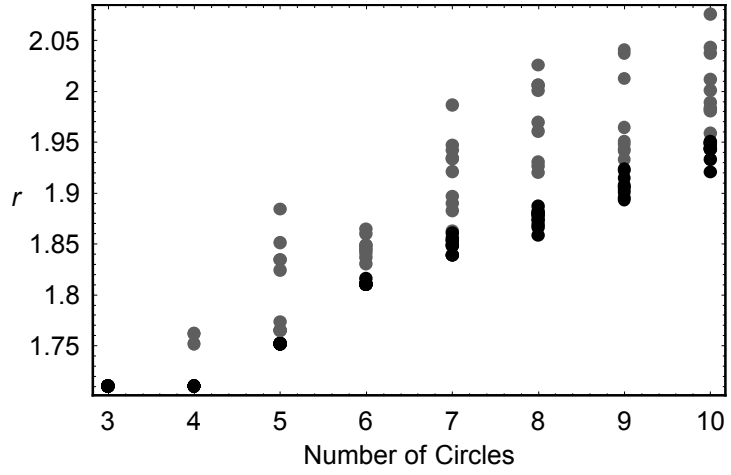

Fig. 5 Plot of circumscribing radius, r, versus number of circles shows the objective function value comparison between MathOptimizer Professional and **NMinimize** for unequal circle packing. The results are based on 10 sequentially performed optimizations, using different random seeds.

The results summarized by the figures show that MathOptimizer Professional ran faster and gave better results than **NMinimize** in the circle packing tests. It should be noted, however, that for simpler and small-size optimization problems with a small number of local minima, **NMinimize** often returns the same result in less time than MathOptimizer Professional. This is due to the 'overhead' incurred in MathOptimizer Professional in compiling the model file, as well as to the *MathLink* based communication between the *Mathematica* notebook and **LGO**. (The **LGO** solver suite runs faster than the total time spent on using the **callLGO** function.)

We also note that MathOptimizer Professional perfomance compares favorably also to that of several other global optimization packages that we have applied to solve non-uniform circle packing models. More detailed comparative results will be presented elsewhere.

6. Concluding Remarks

In this article, we discuss the potentials of *Mathematica* in OR/MS studies, with a special relevance in nonlinear systems modeling and optimization. Following the introduction of the general global optimiza tion model, we present MathOptimizer Professional with an external link to the LGO solver engine.

The usage of MOP is illustrated by solving non-trivial numerical examples. The circle packing models discussed can serve as test models to benchmark global optimization software. Although all numerical test reports depend also on solver parameterizations and other circumstances, the examples presented indicate the capabilities and potentials of MathOptimizer Professional. In this context, we also refer to another recent computational study [44] that compares the performance of the GAMS/LGO implementa· tion to state-of-art local solvers. This evaluation, that also shows the comparative merits of the LGO solver suite, has been done in a fully automated fashion using the GAMS model libraries. Therefore it can be considered as reasonably objective, although the collection of available models and other circumstances always carry elements of arbitrariness and subjectivity. (The details of this performance analysis, including the paper mentioned, are directly available from http://www.gams.com/solvers/ solvers.htm#LGO.)

For over a decade, LGO has been applied in a large variety of research and professional contexts: for examples and case studies see references [27, 29, 45-48]. We expect an essentially similar performance from MathOptimizer Professional that enables the solution of sizeable, sophisticated *Mathematica* models, with an efficiency comparable to that of compiler-based nonlinear solvers.

Acknowledgements

We wish to thank Mark Sofroniou (Wolfram Research) for his permission to use, and to customize, the `Format.m` package in our MOP software development work. We received useful comments and advice from two anonymous referees, as well as from Daniel Lichtblau (Wolfram Research) and Ignacio Castillo (University of Alberta).

JDP's research in recent years has been partially supported by grants received from the National Research Council of Canada (NRC IRAP Project No. 362093) and from the Hungarian Scientific Research Fund (OTKA Grant No. T 034350). JDP also wishes to acknowledge the support received from Wolfram Research over the years, in forms of a visiting scholarship, books, software, and profes- sional advice.

References

[1] S. Wolfram (2003) *The Mathematica Book.* (5th Edition.) Wolfram Media, Champaign, IL, and Cambridge University Press, Cambridge, UK.

[2] *Mathematica* Information Center http://library.wolfram.com/.

[3] P.C. Bell (1999) *Management Science / Operations Research – A Strategic Perspective*. South-West· ern College Publishing / Thomson Learning, Cincinnati, OH.

[4] D. Bertsimas and R.M. Freund, R.M. (2000) *Data, Models, and Decisions*. South-Western College Publishing / Thomson Learning, Cincinnati, OH.

[5] M.A. Bhatti (2000) *Practical Optimization Methods with Mathematica Applications*. Springer, New York, NY.

[6] M.W. Carter and C.C. Price (2001) *Operations Research: A Practical Introduction*. CRC Press, Boca Raton, FL.

[7] U. Diwekar (2003) *Introduction to Applied Optimization*. Kluwer Academic Publishers, Dordrecht.

[8] T.F. Edgar, D.M. Himmelblau and L.S. Lasdon (2001) *Optimization of Chemical Processes*. (2nd Edition.) McGraw-Hill, NY.

[9] F.S. Hillier and G.J. Lieberman (2005) *Introduction to Operations Research*. (8th Edition.) McGraw-Hill, New York, NY.

[10] W.L. Winston and S.C. Albright (2001) *Practical Management Science*. (2nd Edition.) Duxbury / Thomson Learning, Pacific Grove, CA.

[11] INFORMS www.informs.org.

[12] R. Aris (1999) *Mathematical Modeling: A Chemical Engineer's Perspective*. Academic Press, San Diego, CA.

[13] T.B. Bahder (1995) *Mathematica for Scientists and Engineers*. Addison-Wesley, Reading, MA.

[14] R. Gass (1998) *Mathematica for Scientists and Engineers: Using Mathematica to do Science.* Prentice Hall, Englewood Cliffs, NJ.

[15] N.A. Gershenfeld (1999) *The Nature of Mathematical Modeling.* Cambridge University Press, Cambridge, UK.

[16] J.D. Murray (1983) *Mathematical Biology.* Springer, Berlin.

[17] A. Neumaier (1997) Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM Review* 39, 407-460.

[18] P.Y. Papalambros and D.J. Wilde (2000) *Principles of Optimal Design.* Cambridge University Press, Cambridge, UK.

[19] K. Schittkowski (2002) *Numerical Data Fitting in Dynamical Systems.* Kluwer Academic Publishers, Dordrecht.

[20] M. Schroeder (1991) *Fractals, Chaos, Power Laws.* Freeman & Co., New York.

[21] C.A. Floudas and P.M. Pardalos, Eds. (2000) *Optimization in Computational Chemistry and Molecular Biology: Local and Global Approaches.* Kluwer Academic Publishers, Dordrecht.

[22] C.A. Floudas, P.M. Pardalos, C. Adjiman, W.R. Esposito, Z.H. Gumus, S.T. Harding, J.L. Klepeis, C.A. Meyer and C.A. Schweiger (1999) *Handbook of Test Problems in Local and Global Optimization.* Kluwer Academic Publishers, Dordrecht.

[23] I.E. Grossmann, Ed. (1996) *Global Optimization in Engineering Design.* Kluwer Academic Publishers, Dordrecht.

[24] F.J. Kampas and J.D. Pintér (2005) *Advanced Optimization: Scientific, Engineering, and Economic Applications with Mathematica Examples.* Elsevier, Amsterdam. (Forthcoming.)

[25] P.M. Pardalos, D. Shalloway and G. Xue (1996) *Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding.* DIMACS Series, Vol. 23, American Mathematical Society, Providence, RI.

[26] J.D. Pintér (1996) *Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications).* Kluwer Academic Publishers, Dordrecht.

[27] J.D. Pintér (2001) *Computational Global Optimization in Nonlinear Systems: An Interactive Tutorial.* Lionheart Publishing, Atlanta, GA.

[28] J.D. Pintér (2002) Global optimization: software, test problems, and applications. Chapter 15 (pp. 515-569) in: Pardalos and Romeijn, Eds., *Handbook of Global Optimization, Vol. 2.* Kluwer Academic Publishers, Dordrecht.

[29] J.D. Pintér (2005) *Applied Nonlinear Optimization in Modeling Environments*. CRC Press, Boca Raton, FL, 2005. (Forthcoming.)

[30] M. Tawarmalani and N.V. Sahinidis (2002) *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming.* Kluwer Academic Publishers, Dordrecht.

[31] Z.B. Zabinsky (2003) *Stochastic Adaptive Search for Global Optimization.* Kluwer Academic Publishers, Dordrecht.

[32] A. Neumaier (2003) Complete search in continuous optimization and constraint satisfaction. In: Iserles, A., Ed. *Acta Numerica 2004.* Cambridge University Press, Cambridge, UK.

[33] R. Horst, R. and P.M. Pardalos, Eds. (1995) *Handbook of Global Optimization, Vol. 1*. Kluwer Academic Publishers, Dordrecht.

[34] P.M. Pardalos and H.E. Romeijn, Eds. (2002) *Handbook of Global Optimization, Vol. 2.* Kluwer Academic Publishers, Dordrecht.

[35] J.D. Pintér (2005) *LGO – An Integrated Model Development and Solver Environment for Continuous Global Optimization. User Guide.* (Current Edition.) Pintér Consulting Services, Inc., Halifax, NS, Canada.

[36] H.P. Benson and E. Sun (2000) LGO – Versatile tool for global optimization. *ORMS Today* 27 (5) 52-55. Internet version is available at http://www.lionhrtpub.com/orms/orms-10-00/swr.html.

[37] C.G.E. Boender and H.E. Romeijn (1995) Stochastic methods. Chapter 14 (pp. 829-869) in: Horst and Pardalos, Eds. *Handbook of Global Optimization, Vol. 1*. Kluwer Academic Publishers, Dordrecht.

[38] J.D. Pintér and F.J. Kampas (2003) *MathOptimizer Professional – An Advanced Modeling and Optimization System for Mathematica Users with an External Solver Link. User Guide.* Pintér Consulting Services, Inc., Halifax, NS, Canada.

[39] J.B.M. Melissen (1997) *Packing and Covering with Circles.* Ph.D. Dissertation, Universiteit Utrecht, The Netherlands.

[40] E. Specht http://www.packomania.com/.

[41] M.D. Riskin, K.C. Bessette and I. Castillo (2003) A logarithmic barrier approach to solving the dashboard planning problem. *INFOR* 41, 245-257.

[42] Y.Y. Stoyan, G. Yaskov and G. Scheithauer (2003) Packing of various solid spheres into a paralellepiped. *CEJOR* 11, 389-407.

[43] F.J. Kampas and J.D. Pintér (2004) Generalized circle packings: model formulations and numerical results. *Proceedings of the 6th International Mathematica Symposium* (Banff, AB, Canada August 1-6, 2004).

[44] J.D. Pintér (2003) GAMS/LGO nonlinear solver suite: key features, usage, and numerical performance. (Submitted for publication.) Available from http://www.gams.com/solvers/solvers.htm#LGO.

[45] J.D. Pintér (2001) Globally optimized spherical point arrangements: model variants and illustrative results. *Annals of Operations Research* 104, 213-230.

[46] F.J. Kampas and J.D. Pintér (2005) Configuration analysis and design by using optimization tools in *Mathematica*. *The Mathematica Journal* (To appear).

[47] G. Isenor, J.D. Pintér and M. Cada (2003) A global optimization approach to laser design. *Optimization and Engineering* 4, 177-196.

[48] J. Tervo, P. Kolmonen, T. Lyyra-Laitinen, J.D. Pintér and T. Lahtinen (2003) An optimization-based approach to the multiple static delivery technique in radiation therapy. *Annals of Operations Research* 119, 205-227.