Global Optimization in the COCONUT project

Outline of Algorithm
API Design
Inference Engines
Examples

Hermann Schichl,
Arnold Neumaier, Eric Monfroy,
and the COCONUT partners

IST-2000-26063 COCONUT

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

The COCONUT project

- European Union research and development project
- Partners from six European universities:
  Nantes, Lausanne, Vienna
  Louvain-la-Neuve, Coimbra, Darmstadt
  and an industrial partner:
  ILOG
- Aimed at the integration of the existing approaches to continuous global optimization and constraint satisfaction
- December 2000 – November 2003

IST-2000-26063 COCONUT

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Basic Modular Setup

Strategy Eng.

Main part of Algorithm,
makes decisions

Inference Eng.

Problem structure analysis,
local optimization,
constraint satisfaction, interval
analysis, Linear relaxation,
convex opt., bisection,...

Reports

Produces files (AMPL),
human readable output,
checkpointing

Management

Problem management,
Resources, Parallelization

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Modular API design

- The API is designed to make the development of the various module types independent of each other and independent of the internal model representation.
- A collection of C++ classes.
- Uses FILIB++ and MTL.
- Supports dynamic linking.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

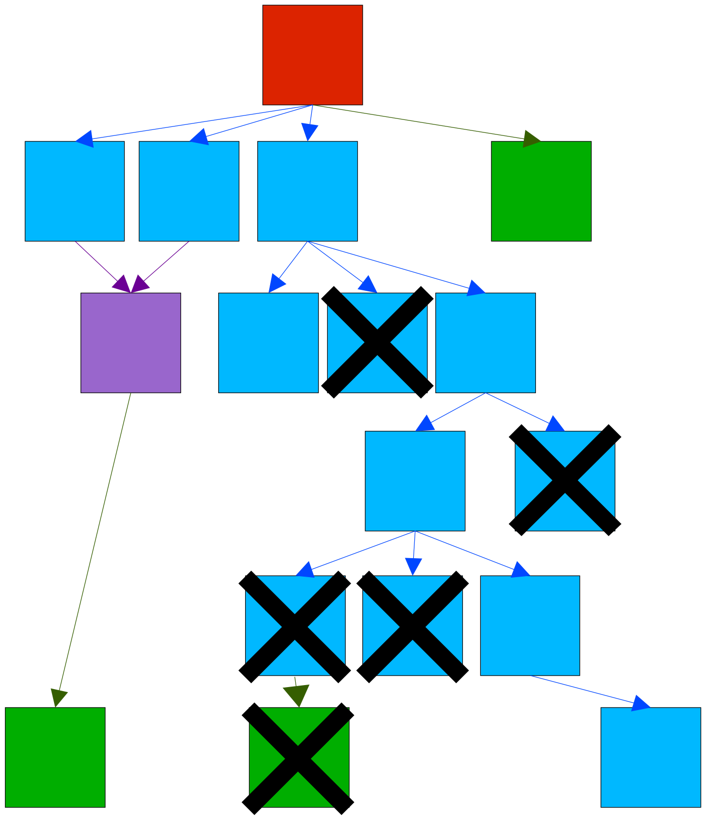
Modular API design (cont.d)

- All inference engines are subclasses of one class, so they have the same basic structure.
- Communication with the strategy engine by a database-like communication.
- The API implementation (w/o inference engines) consists of **44000** lines of C++ code and a few perl scripts, organized into 128 files, occupying **1.3 MB** of disk space.

IST-2000-26063 COCONUT

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Search graph



- Starts at the **original model**
- Contains **relaxations**
- and **splits**.
- It is not a tree since it might also contain **glueings**.
- Some of the nodes will be **terminal**, since they can be solved completely.

Model Relations

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & a(x) \le 0 \\ & x \in [x] \end{aligned}$$

Relaxation

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & x \in [x] \end{aligned}$$

Reduction e.g. Add cut, prune box

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & d^\top x \le d_0 \\ & a(x) \le 0 \\ & x \in [x] \end{aligned}$$

Split

$$[x'] \cup [x''] = [x]$$

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & a(x) \le 0 \\ & x \in [x'] \subset [x] \end{aligned}$$

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & a(x) \le 0 \\ & x \in [x''] \subset [x] \end{aligned}$$

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Model Reductions

$$
\begin{aligned}
\min \quad & c^\top x \\
\text{s.t.} \quad & b^\top x \le b_0 \\
& a(x) \le 0 \\
& x \in [x]
\end{aligned}
$$

Relaxation

$$
\begin{aligned}
\min \quad & c^\top x \\
\text{s.t.} \quad & b^\top x \le b_0 \\
& x \in [x]
\end{aligned}
$$

Split $[x'] \cup [x''] = [x]$

$$
\begin{aligned}
\min \quad & c^\top x \\
\text{s.t.} \quad & b^\top x \le b_0 \\
& a(x) \le 0 \\
& x \in [x'] \subset [x]
\end{aligned}
$$

$$
\begin{aligned}
\min \quad & c^\top x \\
\text{s.t.} \quad & b^\top x \le b_0 \\
& a(x) \le 0 \\
& x \in [x''] \subset [x]
\end{aligned}
$$

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Model Glueing

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & a(x) \le 0 \\ & x \in [x] \end{aligned}$$

Glue, Unsplit

$$[x'] \cup [x''] = [x]$$

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & a(x) \le 0 \\ & x \in [x'] \subset [x] \end{aligned}$$

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & a(x) \le 0 \\ & x \in [x''] \subset [x] \end{aligned}$$

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Internal Representation

- Models are organized in the **search graph**, represented by a Directed Acyclic Graph (DAG).
- For every model in the search graph the following information is stored:
  - Every equation/inequality is assigned a number of **annotations** describing its properties (e.g. linear, quadr., convex, separable, redundant, ...).
  - Additional **local information** (e.g. local optima, active constraints, Lagrangian multipliers,...) is added.
  - A description of the **relation** between the problem and its parent is provided.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Search graph implementation

- The DAG is implemented using the VGTL, a library following the generic programming spirit of the C++ STL.
- There are two types of nodes:
  - **Full nodes** contain complete descriptions of models,
  - **Delta nodes** contain only the changes to the parent model in order to save storage capacity
- The search graph has a **focus** pointing to the model which is worked upon. This model is copied into an enhanced structure – the **work node**. A reference to this work node is passed to the inference engines.
- The graph itself can be analyzed by **search inspectors**.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Internal Mathematical Representation

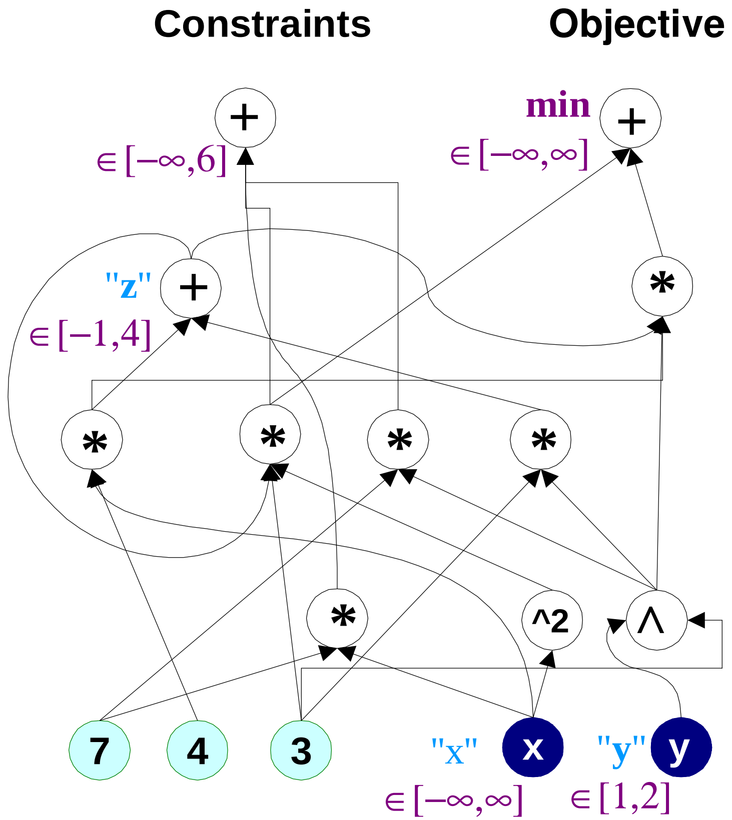
- The internal mathematical representation of a problem is

$$
\begin{aligned}
\min\quad & f_{lin}(x) + f_{quad}(x) + f_{sep}(x) + f_0(x) \\
s.t.\quad & G_{lin}(x) + G_{quad}(x) + G_{sep}(x) + G_0(x) \in S_c \\
& x \in S_v
\end{aligned}
$$

  where (currently) the sets S are boxes.
- The algorithmic representation is in graph form using not a tree (or forest) as usual but a **directed acyclic graph**.
- Variables appearing left of an assignment are substituted out.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Directed Acyclic Graph (DAG)



- DAG representation of the model

$$\min \quad 3x^2z + 4xy^3z$$

$$s.t. \quad z = 4x + 3y^3$$

$$7x + 3x^2z + 7y^3 \le 6$$

$$y \in [1,2], z \in [-1,4]$$

- similar to computational tree
- every node is an expression
- a node may have more than one parent
- Constants and variables are sources, objective and constraints are sinks

Expressions

- Every vertex represents a function *F* mapping a vector $x \in \mathbb{R}^n$ to a value $F(x) \in \mathbb{R}$.
- Predefined functions include **sum**, **product**, **max**, **min**, elementary real functions (**exp**, **log**, **pow**, **sqrt**, ...)
- **Variable indicator** contains the indices of the variables this vertex depends on.
- Additional information is added (ranges, semantics, variable name, vertex number, ...)

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Evaluation of a DAG

- Evaluation works similar to computation trees by performing a **graph walk**.
- **Caching** keeps evaluation work minimal.
- The whole model is stored in **one** graph.
- Defining **short-cuts** makes it possible to replace graph walks by evaluation functions. Short-cuts may be defined at every node.
- Additional elementary functions can easily be incorporated.

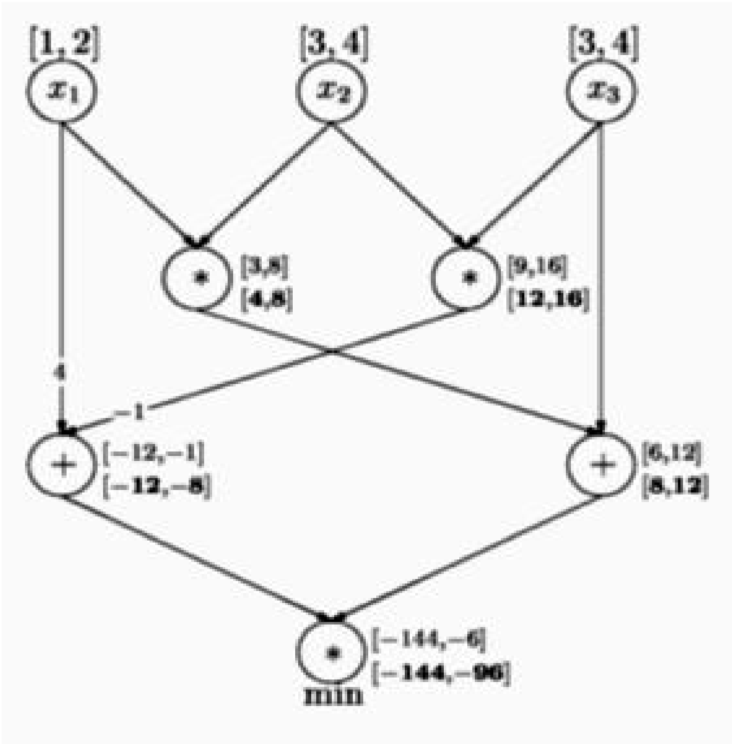
Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Graphs and Evaluators

- Generic Graph Library (**V**ienna **G**raph **T**emplate **L**ibrary) in C++ to construct and manipulate DAGs, and forests (trees).
- Generic Programming approach with **containers**, **walkers**, **function objects**, and **generic algorithms**.
- For expression graphs (DAG or tree) special visitors are provided — **(cached) forward and backward evaluators**.
- Currently implemented Evaluators:
  - Real Function Values and Function Ranges
  - Gradients (Real, Interval)
  - Slopes
- In the near future Evaluators for:
  - Hessians (Real, Interval)
  - Second order Slopes

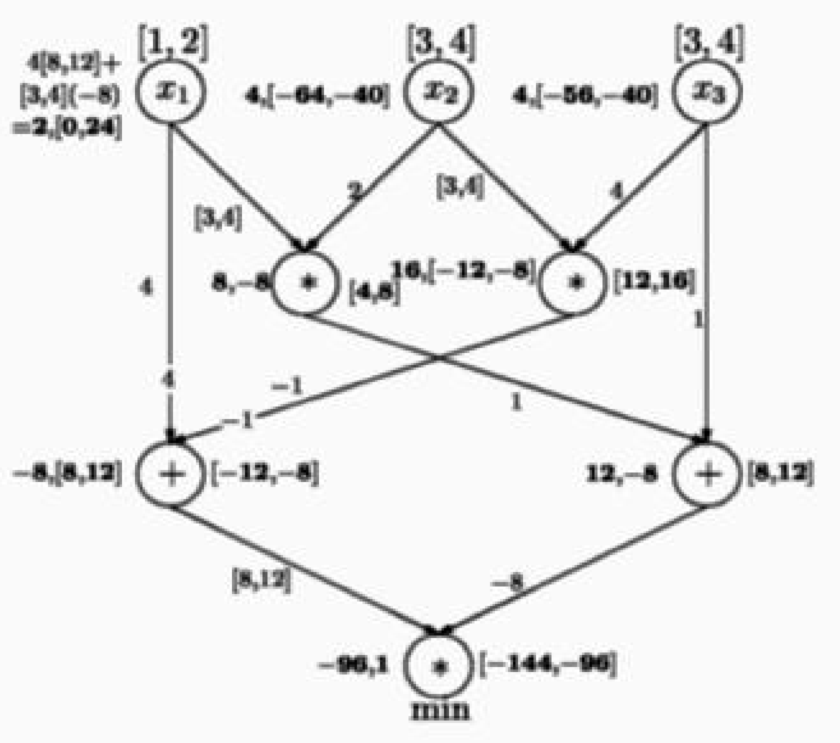
Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Example



- **Interval evaluation and constraint propagation produce bounds on each node**
- **No reduction on the domain of the variables**
- **The bounds on intermediate nodes are improved compared to interval evaluation**

Example (ctd.)



- **Linear enclosures produced using slopes give redundant constraints, e.g.**

$$24(x_1-2)-48(x_2-4)-32(x_3-4)\le 0$$

Example (ctd.)

- Now constraint propagation leads to a reduction of the domain of the variables

$$x_2 \in [3.4, 4]$$
$$x_3 \in [3.4, 4]$$

- With previously known techniques but without (expensive) higher order consistency, such a reduction would have required a split of the box.

Inference Engines

- Corresponding to every type of problem change, a class of **inference engines** is designed:
  - Model analysis (e.g. find convex part)
  - Model reduction (e.g. pruning, fathoming)
  - Model relaxation (e.g. linear relaxation)
  - Model splitting (e.g. bisection)
  - Model glueing (e.g. undo excessive splitting)
  - Update local information (e.g. probing, local optimization)
  - Check certificate (check correctness of calculation)
- Inference engines **never change** the model but calculate which changes may be made and are considered useful.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engines: General features

- All inference modules only advertise changes.
- There is a fixed documentation structure defined.
  - Services Provided
  - Limits
  - Structure, Prerequisites of Input
  - Structure, Features of Output
  - Control Parameters
  - Termination Reason
- They produce a **result** where every possible change is listed together with a **weight** (the higher the weight the more important the change) and a **certificate** for correctness.
- They collect **statistical data** to support the strategy engine in making decisions.

IST-2000-26063 COCONUT

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engines implemented as State of the Art

- Several **state of the art techniqnes** were implemented as **inference engines**:
  - STOP (starting point generator)
  - DONLP2-INTV (local optimizer)
  - Karush-John-Condition Generator
  - Point Verifier
  - Exclusion Box
  - Interval constraint propagation
  - Linear Relaxation
  - CPLEX (linear programming solver)
  - Basic Splitter
  - BCS (box covering solver)

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engine: STOP

- **Heuristic** Global Optimization Algorithm
- Combines Multi-Level-Coordinate-Search and Constraint Propagation
- Produces Starting Points for Local Optimization

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engine: DONLP2-INTV

- General purpose non-linear local optimizer for continuous variables
- SQP method
- Dense Linear Algebra
- Envelope uses standard evaluators, gradients are computed by automatic differentiation

Inference Engine: Karush-John Conditions

- Generates the DAG representation of the Karush-John first order optimality conditions
- Every constraint (even two-sided) gets associated one Lagrange multiplier
- Constructed by symbolic differentiation of the DAG representation

IST-2000-26063 COCONUT

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engine: Point Verifier

- Computes a **uniqueness region** around an approximate solution, in particular a **verified point**
- Uses Karush-John conditions
- Tries to maximize the uniqueness region by inflation

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engine: Exclusion Box

- Derives a large **exclusion box** and a tiny **inclusion box** such that the area between these two boxes does not contain a local optimizer.
- They are computed around an approximate local optimizer to **get rid of the cluster effect**.
- Does not focus on uniqueness.
- Uses slopes and H-matrix techniques.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engine: Constraint Propagation

- Performs the hull-consistency algorithm for **constraint propagation**.
- Reduces the possible range of the variables
- Might detect infeasibility of the problem

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engine: Linear Relaxation

- Computes a linear relaxation of the problem.
- Uses centered forms and slopes to compute the linear inequalities.
- Makes use of the DAG enhancements to improve the slopes.
- Either adds the linear relaxation as cuts or generates a full linear model.

IST-2000-26063 COCONUT

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engine: CPLEX

- Solves linear problems.
- Interfaces the state-of-the-art commercial linear solver CPLEX.
- Extremely good performance.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engine: Basic Split

- Provides **splitting coordinates** and **split points**.
- Computes a difficulty estimate for the variables involved.
- Suggests splits for the *n* most difficult variables.
- Uses exclusion box and solution information to improve the choice of split points.
- Cuts exclusion boxes out of the search area by careful choice of splitting coordinates.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Inference Engine: BCS

- Covers the feasible area by boxes.
- Uses DMBC (dichotomous maintaining bound-consistency) and UCA6 (union-conservative approximation) in both basic and enhanced variants.
- Distinguishes between boxes in the interior and at the border of the feasible region.
- Uses the commercial ILOG Solver, or the constraint propagator provided by IRIN, but can work with any constraint propagator

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

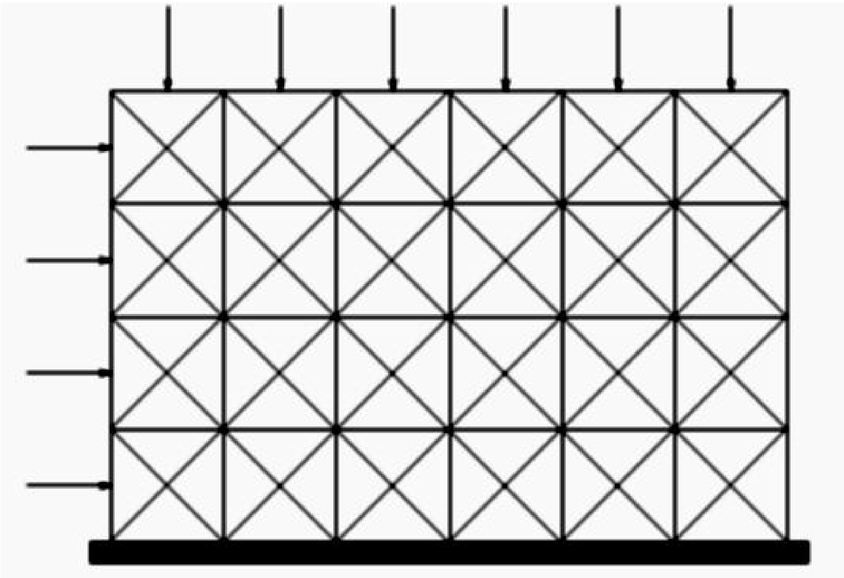
Contributions from the outside of the COCONUT project

We are happy that researchers and companies from outside the COCONUT project agreed to complement our efforts in integrating the known techniques:

- Bernstein modules by J. Garloff & A. Smith (U. Konstanz)
- Verified lower bounds for convex relaxations by Ch. Jansson (TU Hamburg-Harburg)
- GAMS reader by the GAMS consortium
- Taylor arithmetic by G. Corliss (Marquette U.)
- Asymptotic arithmetic by K. Petras (U. Braunschweig)
- XPRESS commercial LP-solver (Dash Optimization)
- Hopefully additional contributions by you!

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Worst case finite element analysis



- Linear FEM equations become non-convex if material data is uncertain.
- Typical size of uncertainty is 10-20% in elasticity and cross-section area.
- Law requires the computation of the worst case.
- Industry relevant problems have some thousand variables.

Worst case FEM structural analysis
Promising result

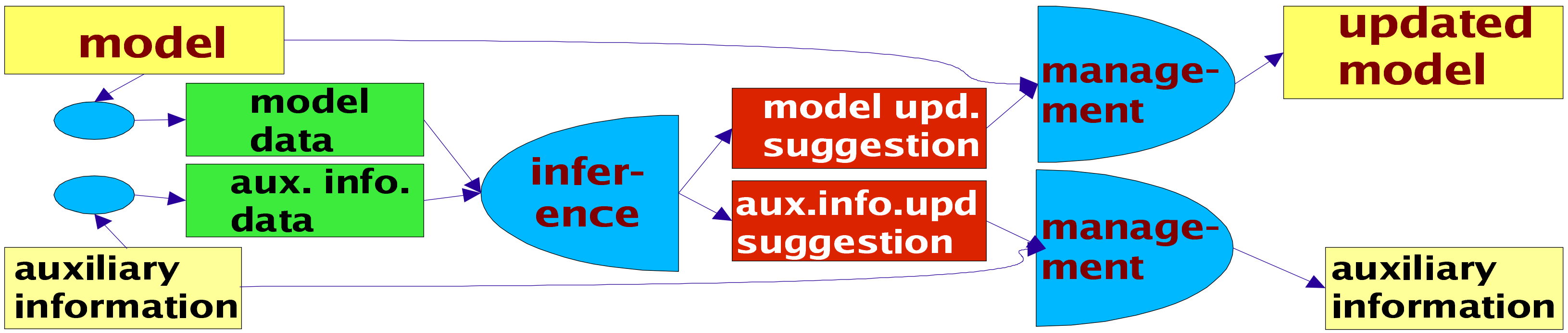
- **Worst case analysis on the displacements *u* for a 20x20 wall in the non-linear system**

$$A(x)u = b$$

- **1620 material parameters *x* with 16.4% uncertainty, 840 displacements *u***
- **Traditional methods fail for 0.01% uncertainty**
- **Exploiting the special structure, within 30s on a 1.6 Ghz Pentium 4, without bisection we get**

Uncertainty (%)	0.01	0.05	0.5	1	2.5	5	10	16.4
Overestimation	1.03	1.15	2.55	4.12	8.92	17.26	35.33	61.59

Basic algorithm design



- This setup allows for highest flexibility and extensibility
  - the modules are split into inference engines (calculation) and management parts
  - additional modules for model handling are added
- The strategy engine decides which components are called in every algorithmic step of this type

Report Modules

- This class of modules produces **output**.
  Various types of files and human readable output will have to be created.
- Examples:
  - Solution Report (humans, AMPL, GAMS)
  - Progress Information
  - Checkpointing
  - Debugging Information
  - Error Messages

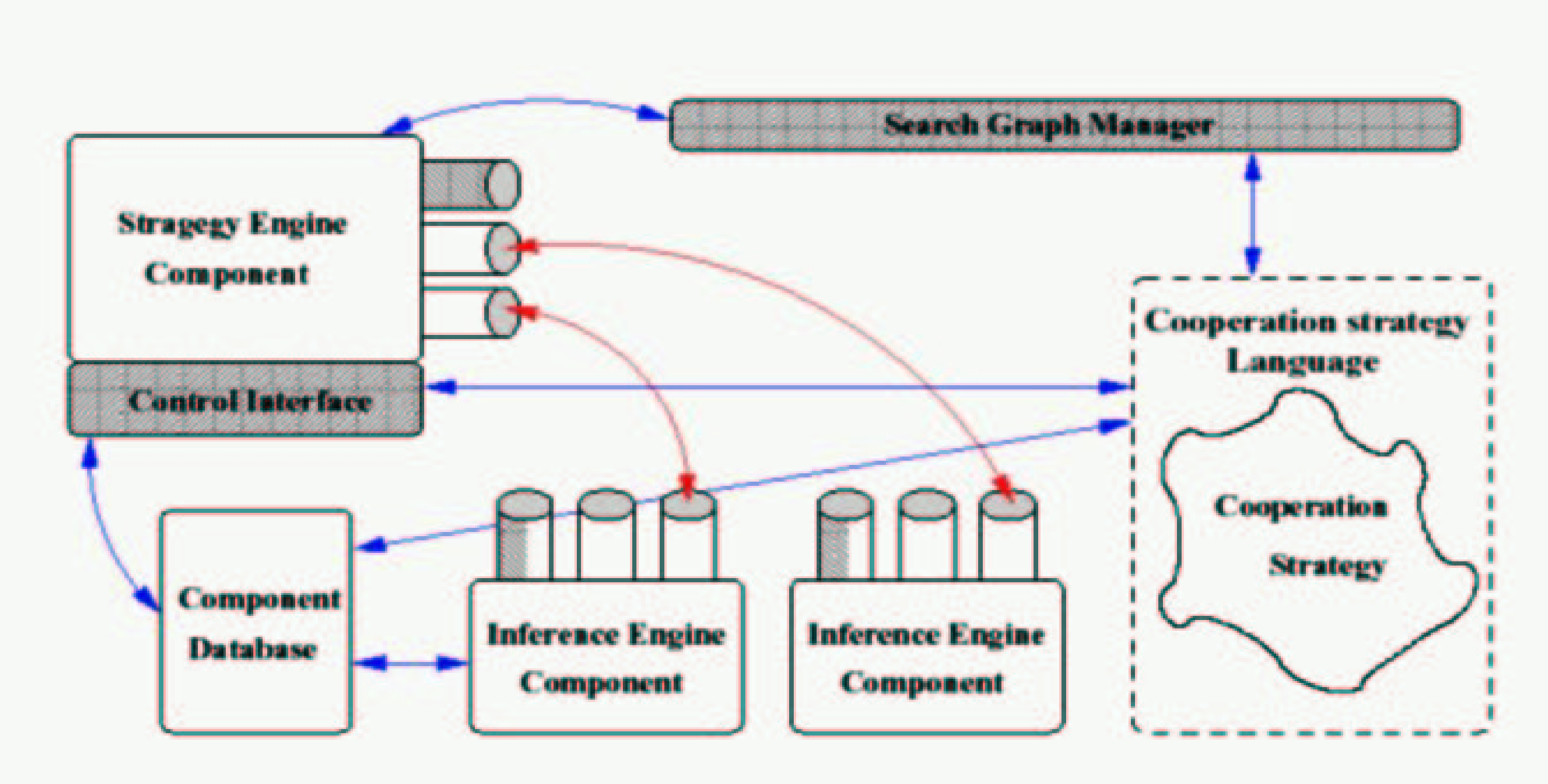
Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Management Modules

- Corresponding to every internal part of the program, a class of **management modules** is designed:
  - Model management
  - Data collection
  - Resource management
  - Initialization management
- Management modules **never calculate** anything. They just **perform some** of the changes which have been advertised by inference modules.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Strategy Engine



Strategy Engine (ctd.)

- It is the core of the algorithm and consists of
  - The **logic core** ("search") which is essentially the main solution loop,
  - Special **decision makers** (very specialized inference engines) for determining the next action at every point in the algorithm.
- It calls the management modules, the report modules, and the inference engines in succession.
- It can be programmed using a simple **strategy language** (interpreted, Python based).
  - (Semi-)interactive and automatic solution process
  - Debugging and single-stepping of strategies
  - Object oriented, dynamically typed objects, garbage collected
  - Easily extendable

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Strategy Engine (ctd.)

- Manages the search graph via the **search graph manager**,
- Manages the search database via the **database manager**,
- Uses a **component framework** to communicate with the inference engines,
- Launches inference engines dynamically (on need) to avoid memory overload,
- Provides a management interface,
- Strategy engine is itself a component, so multilevel strategies are possible,
- Prepared for **distributed and parallel computing**, and **distributed memory**

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Extensibility

- The strategy language makes it easy to change the strategy.
- A registration phase during initialization removes the need to recompile the program when new inference engines are added.
- Registration also allows us to balance scientific and commercial interests:
  - Free but reduced core version with open API specification
  - Free strategy engine with basic strategy
  - Advanced commercial components
- Extending the system by external contributers is made easy by this modular design.

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

Invitation

We hope that the community will contribute to this algorithmic framework.

The End

Thank you for your attention!

COCONUT Website:

http://www.mat.univie.ac.at/coconut

IST-2000-26063 COCONUT

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme