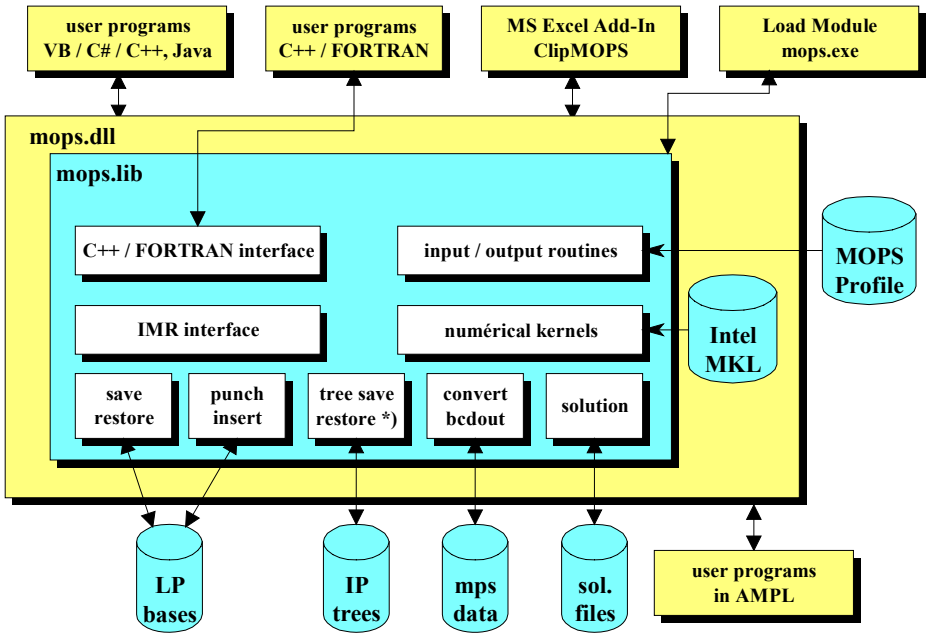
# Kapitel 9

# Einführung in MOPS
# API-Funktionen

Uwe H. Suhl
Lehrstuhl für Wirtschaftsinformatik
Freie Universität Berlin

Optimierungssysteme
Version 1.1 / SS 2008

## MOPS - External Architecture



## Einbettung von MOPS API-Funktionen in EUS Software

**Four principal possibilities (each in 32 Bit and 64 Bit Addressing)**

- **MOPS load module: *mops.exe. Mops64.exe***
  - Windows character executable
  - Data input format: mps and triplet format
  - Mops profile allows customizing of optimization
- **MOPS callable object code library: *mops.lib, mops64.lib***
  - Allows access to all internal data und functions
  - Model can be generated in internal arrays of optimizer
  - One address space with application possible
  - MOPS functions can only be called from compatible systems such as MS Visual C++ or Intel C++ / FORTRAN (VS6, VS2003, VS2005)
- **MOPS Dynamic Link Library: *mops.dll, mops64.dll***
  - DLL functions can be called from nearly all Windows programs such as Visual Basic (.NET), C#, Delphi. Java, C++ etc.
  - Separate address spaces of application and optimizer
  - Easy integration: no link step, just replace the dll if calling sequence of functions are not changed
- **MOPS Studio**
  - Offers interactive modelling and optimization based on AMPL, Math. Progr. Lang.
  - Windows based GUI and editor for AMPL, optimization and solutions

## Main MOPS Dll functions

- **Initialize ()** initialization for a new model, resets parameters and values
- **PutModel (intyp, inf, m, n, nz, ia, ja, a, lb, ub, c, typ)** passes a model to the MOPS Dll 1. as triplets, 2. rowwise 3. columnwise in arrays ia,ja,lb,ub,c,typ and allocates memory for model and temporary data
- **Optimize (dir,status,phase,funct)** solves an LP / IP model where dir is the optimization direction and the other parameters are output
- **SetParameter (s)** where s is a string containing parameter assignments
- **GetParameter (Parameter, value)** retrieves the value of a MOPS parameter
- **GetLPSolution (lpsta, lpFunct, Activity, RedCost, Status)** stores status and values of the initial LP solution in user given arrays
- **GetIPSolution (ipsta, ipFunct, Activity, RedCost, Status)** stores status and values of an IP-solution in user given arrays
- **Finish ()** deallocates the memory block and closes all files
- **Many more functions for model handling!**

# Beispiel Lösung von Sudoku-Rätseln (→ Demo)

- **Benutzte Funktionen in VBA unter Excel**

```
Declare Function FreeMemory Lib "mops.dll" () As Long
Declare Function LoadLibrary Lib "kernel32" Alias "LoadLibraryA" (ByVal FileName As String) As Long
Declare Function GetIPSolution Lib "mops.dll" (lipsta&, dLPfunct#, dxs#, ddj#, lsta&) As Long
Declare Function Initialize Lib "mops.dll" () As Long
Declare Function Optimize Lib "mops.dll" (ByVal ldirection&, lstatus&, lphase&, dZf#) As Long
Declare Function PutModel Lib "mops.dll" (ByVal lintyp&, ByVal inf#, ByVal m&, ByVal n&, ByVal nz&, ia&, ja&,
      a#, lb#, ub#, c#, typ&) As Long
Declare Function SetParameter Lib "mops.dll" (ByVal s$) As Long
```

- **Prinzip der Lösung in VBA**

```
'PutModel Arrays für die Aufnahme des Modells
Dim intyp As Long, inf As Double, m As Long, n As Long, nz As Long
Dim ia(1 To 10000) As Long, ja(1 To 10000) As Long, a(1 To 10000) As Double
Dim lb(1 To 5000) As Double, ub(1 To 5000) As Double, c(1 To 5000) As Double
Dim colType(1 To 5000) As Long, k As Long; Dim ColName1(1 To 5000) As Long
Dim ColName2(1 To 5000) As Long; Dim ColName3(1 To 5000) As Long
```

**…. Generate model in PutModel arrays**

```
rc = LoadLibrary("c:\programme\mops\mops dll\mops.dll")  # Pfad zur MOPS Dll
rc = Initialize() # Initialisierung für ein neues Modell
rc = PutModel(intyp, inf, m, n, nz, ia(1), ja(1), a(1), lb(1), ub(1), c(1), colType(1))  # Übergabe des Modells an MOPS
rc = Optimize(1, xipsta, xphase, xzbest) # Minimierung des Modells
rc = GetIPSolution(xipsta, xzbest, Activity(1), RedCost(1), ColStatus(1)) # Auslesen der Integer Lösung für VBA
Rc = Finish()
```