在 **win10/11** 下安装 **WSL2+**编译 **SCIP** 套件

笔者以 win11+Ubuntu22.04 为例

前言：

SCIP Opt Suite(https://scipopt.org/#scipoptsuite)是德国 ZIB(Zuse Institute Berlin)研究所开发的混合整数规划求解套装，可以求解 Pseudoboolean optimization，Satisfiability (SAT) and variants，CIP，MILP，不论凸还是非凸的 MINLP 以及其连续问题。该求解套装由 SCIP 分支定界框架+Soplex LP 求解器+ZIMPL 建模语言+UG 分布计算框架+GCG 通用列生成求解器构成，非线性求解器需要自行安装开源的 IPOPT 或者不开源的 WORHP 求解器，LP 求解器也可以在编译时替换成 Cplex、Gurobi、Xpress 等商业求解器以提升效率。该求解器对各类优化问题和模型文件有着极好的兼容性，包括但不限于 AMPL 的 nl 格式模型，Cplex 的 lp 格式模型和通用的 mps 格式模型等，并有原生的 C/C++ API 和 Python API。

特别需要指出的是：经过几个大版本的迭代，GCG 求解器已经具备了自动化和图示化的 Dantzig-Wolfe 分解和 Benders 分解功能，但是该求解器基于 GNU 开发，对 Linux 和 Unix 系统支持很好，不支持 Windows 系列，拉高了使用者的门槛。笔者同时拥有 Mac、Ubuntu 和 Window 电脑，以 Mac 电脑为办公主力，但是受限于 Mac 系统金子般的内存颗粒价格和分解问题巨大的内存开销，转而将求解转移到了 Ubuntu 上，但是 Ubuntu 不会是大多数人的主力操作系统，笔者也深感不变，后改为使用 WSL2+Ubuntu 方式，放在 Windows 电脑上，获得了非常好的体验，所以编写了这份教程。

最优化问题对处理器单核 ipc 和内存大小及带宽有极大需求，非常不推荐低配电脑使用该教程，会获得非常糟糕的体验，推荐配置：Windows11 专业版+32GB 以上 RAM+12 代及以后的 Intel 酷睿处理器+50G 以上的硬盘空间

WSL2 是 Windows Subsystem for Linux 的简称，可以实现 Linux 系统二进制文件在 Windows 上的转译执行，姑且就把它当成虚拟机吧，WSL2 的执行效率高于 VM、PD 等虚拟机软件，硬件开销也不大，能够有原生物理机的 9 成以上功力。

1.首先开启 win10/win11 下的虚拟机平台和适用于 Linux 的 windows 子系统，以管理员身份运行 Windows Powershell，输入以下代码：

适用于 Linux 的 Windows 子系统

dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

虚拟机平台
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

如果本机有多台虚拟机，为了安全可以开启 Hyper-V，具体步骤请自行搜索，不影响主体功能。

以上功能开启后需重新启动，随后输入：

获取现有 wsl 版本信息

wsl --list —verbose

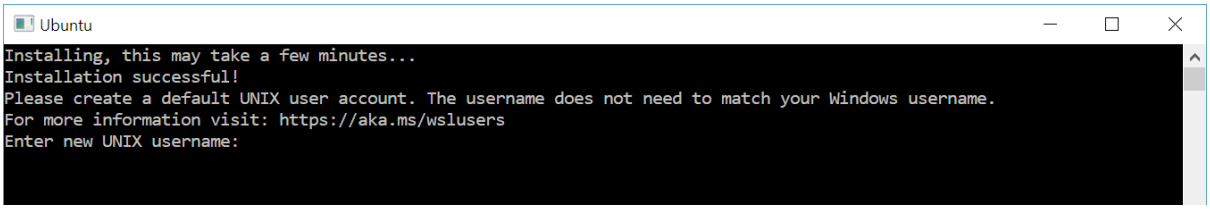
#开启管理员权限

wsl --set-default-version 2

2.重启完毕之后进入 windows 商店，搜素 Ubuntu22.04，点击下载，随后会自动安装。

下载完成后，点击图标进入，首次使用会需要几分钟执行安装。

完成安装后，设置 Ubuntu 用户名（非 root）及密码，正式开启 Ubuntu 系统。



3.将你的 Ubuntu22.04 实例移动到其他磁盘(只有 C 盘可以跳过该节)

首先将你现在的 Ubuntu 实例导出,以在 D 盘新建 Linux 文件夹为例：

wsl --export Ubuntu-22.04 D:\Linux\ubuntu-22.04.tar

注销 Ubuntu 22.04 实例：

wsl --unregister Ubuntu-22.04

导入到新位置：

wsl --import Ubuntu-22.04 D:\Linux D:\Linux\ubuntu-22.04.tar

验证迁移：

wsl -d Ubuntu-22.04

验证完毕后在 Windows Powershell 输入 wsl，就进入了 Ubuntu22.04，很抱歉该系统暂时没有支持图形界面的 wsl 镜像，如果对图形界面有刚性需求的请安装 Ubuntu20.04-wsl 镜像版，该版本有图形界面，本文不做展开。

需要注意的是该系统较老，apt 等程序还没有更新，后续装软件时会报错，请首先更新 apt 并修复问题：

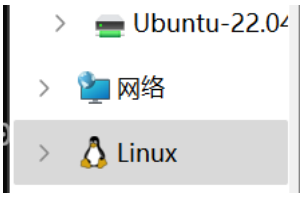
sudo apt-get update

sudo apt-get install -f —fix-missing

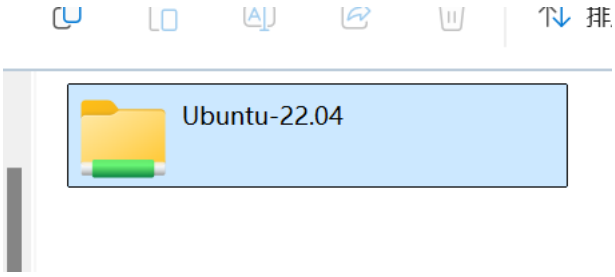
此时你的 Ubuntu22.04-WSL 已经准备就绪，同普通的 Ubuntu 无异。

4.实现 Linux 和 Windows 的文件互相管理

完成以上工作后，就可以将 Ubuntu 实例安装到 D 盘之外的地方，此时在我的电脑内可以看到



点击 Linux 的图标



右键 Ubuntu-22.04 文件夹，选择映射为网络驱动器，一路确认后，你的 Linux 文件夹就变成了电脑的 Z 盘，可以和 Windows 直接跨系统处理文件了。

*在 WSL 系统内输入 gcc -dM -E -march=native - < /dev/null | egrep "SSE|AVX" | sort，如果是 clang 编译器就把 gcc 改成 clang，笔者强烈推荐 clang 编译器。会显示支持的指令集，搜索一下是否有 AVX2，后续编译软件过程，笔者都启用了编译器的自动编译优化，面向 AVX2 指令集向量化编译优化，如果是 intel 酷睿 4 代之前的处理器，请改成 AVX 或者其他兼容的指令集。（可选）

```
(base) root@bbbb2210:/mnt/c/Users/shengtianyi# clang -dM -E -march=native - < /dev/null | egrep "SSE|AVX" | sort
#define __AVX2__ 1
#define __AVXVNNI__ 1
#define __AVX__ 1
#define __SSE2_MATH__ 1
#define __SSE2__ 1
#define __SSE3__ 1
#define __SSE4_1__ 1
#define __SSE4_2__ 1
#define __SSE_MATH__ 1
#define __SSE__ 1
#define __SSSE3__ 1
```

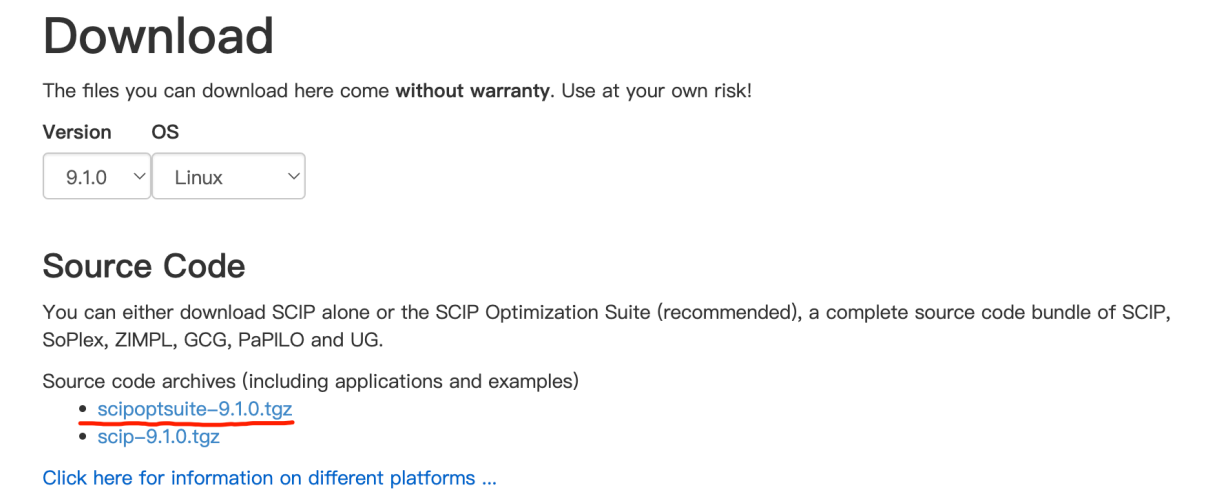
5.申请 Cplex 等求解器的 Linux 学术版(可选)

Cplex 等求解器可以作为 SCIP 的底层 LP 求解器，相比自带的开源求解器 Soplex 有一定提升，小规模问题能提升 10%左右，大规模问题可靠性则远胜 Soplex。笔者申请了学术版的 Linux 版 Cplex 并安装在了默认位置。关于求解器安装本人不做讲解，请自行咨询软件客服。

6.下载编译 SCIP 源代码

网址：https://www.scipopt.org/index.php#download

选择 scipoptsuite-9.1.0.tgz：



按照要求填信息下载解压，并移动到 Linux 系统的文件夹内，请自行管理，本人安装在/opt/文件夹下。

安装必要的组件：

apt-get install wget cmake g++ m4 xz-utils libgmp-dev unzip zlib1g-dev libboost-program-options-dev libboost-serialization-dev libboost-regex-dev libboost-iostreams-dev libtbb-dev libreadline-dev pkg-config git liblapack-dev libgsl-dev flex bison libcliquer-dev gfortran file dpkg-dev libopenblas-dev rpm cliquer gnuplot

6.1 Ipopt 非线性求解器的安装(难点)

首先安装 Ipopt 的几个必备库

sudo apt-get install gcc g++ gfortran git patch wget pkg-config liblapack-dev libmetis-dev

接下来安装 ASL(Ampl Solver Library)，MUMPS Linear Solver，HSL (Harwell Subroutines Library)，特别是 HSL ，他是 Ipopt 的线性求解库，该处为最难点，请务必按照介绍安装

先前往 https://licences.stfc.ac.uk/product/coin-hsl

使用学术邮箱注册，并申请 coinhsl-archive-2022.12.02 下载，更新的版本使用 meson 构建不稳定，并且与后续的包不兼容

下载 coinhsl-archive-2022.12.02.tar.gz 文件，解压到 coinhsl 文件夹，暂时放在那。

然后前往/opt/文件夹，使用 git 下载 ThirdParty-HSL：

git clone https://github.com/coin-or-tools/ThirdParty-HSL.git

cd ThirdParty-HSL

回到 coinhsl 文件的根目录，将刚才的 coinhsl 整体复制到当前文件夹，即 ThirdParty-HSL：

cp -r coinhsl /opt/ThirdParty-HSL

然后进入复制后的 coinhsl 文件夹，进行 HSL 构建：

./configure CFLAGS='-O3 -mavx2' FFLAGS='-O3 -mavx2' CXXFLAGS='-O3 -mavx2'

make

sudo make install

安装安装完成后退回上级 ThirdParty-HSL 文件夹，进行完全构建：

./configure CFLAGS='-O3 -mavx2' FFLAGS='-O3 -mavx2' CXXFLAGS='-O3 -mavx2'

make

sudo make install

这两步如果都不报错，则 HSL 就完整的构建好了。

下面构建 ASL 和 MUMPS，这两个都不难。先在/opt/文件夹构建 ASL：

git clone https://github.com/coin-or-tools/ThirdParty-ASL.git

cd ThirdParty-ASL

./get.ASL

./configure CFLAGS='-O3 -mavx2' FFLAGS='-O3 -mavx2' CXXFLAGS='-O3 -mavx2'

make

sudo make install

然后还是在/opt/文件夹构建 MUMPS：

git clone https://github.com/coin-or-tools/ThirdParty-Mumps.git

cd ThirdParty-Mumps

./get.Mumps

./configure CFLAGS='-O3 -mavx2' FFLAGS='-O3 -mavx2' CXXFLAGS='-O3 -mavx2'

make

sudo make install

接下来构建 Ipopt，首先在/opt/文件夹下 git 克隆：

git clone https://github.com/coin-or/Ipopt.git

然后就进去安装就行：

cd Ipopt

mkdir build

cd build

../configure CFLAGS='-O3 -mavx2' FFLAGS='-O3 -mavx2' CXXFLAGS='-O3 -mavx2'

make

make test

make install

安装过程请注意有无报错信息，安装完毕后务必更新动态链接库 sudo ldconfig。

6.2 SCIP 安装

进入/opt/scipoptsuite-9.1.0/文件夹下

首先找到 CMakeLists 文件，移动到 Win11 后在第一行 cmake_minimum_required(VERSION 3.9)下方加入插入一段：

set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -mavx2 -O3")

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -mavx2 -O3")

option(BUILD_PARALLEL "Enable parallel build" ON)

if(BUILD_PARALLEL)

  set(CMAKE_BUILD_PARALLEL_LEVEL 6) # 设置默认的并行线程数，可以修改

endif()然后再复制替换原位的 CMakeLists 文件。如果会用 nano，vim，emacs 等编辑器的可以直接原位修改：

```
cmake_minimum_required(VERSION 3.9)

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -mavx2 -O3")
set(CMAKE_C_FLAGS "${CMAKE_CXX_FLAGS} -mavx2 -O3")
set(CMAKE_VERBOSE_MAKEFILE ON)

option(BUILD_PARALLEL "Enable parallel build" ON)
if(BUILD_PARALLEL)
   set(CMAKE_BUILD_PARALLEL_LEVEL 8) # 设置默认的并行线程数，可以修改
endif()
```

以上代码作用是编译时进行 O3 优化提高 SCIP 的执行效率，而 O3 优化编译相当耗时，所以开启并行编译。

之后在当前文件夹下建立 build 文件夹：

#建立 build 文件夹

mkdir build

cd build

进入 build 文件夹后就可以开始 cmake 编译的过程。有必要说明一下笔者不需要其非线性求解功能，所以没有选装非线性求解器 IPOPT，编译时需要将该功能关闭，如果有非线性需求，请安装 IPOPT。

如果你顺利的安装了 Ipopt 在该文件夹下输入：

cmake .. -DLPS=cpx -DTPI=tny

如果你被 IPOPT 整崩溃了就别装了：

cmake .. -DIPOPT=false -DLPS=cpx -DTPI=tny

然后便可以自动寻找编译需要的组件，如果提示 cpx 功能有错误，原因是没有找到 cplex 的路径，可以在/opt/scipoptsuite-9.1.0/gcg/cmake/Modules 文件夹下找到 FindCPLEX.cmake 文件，打开并在第一行加入并保存：

set(CPLEX_DIR = "/opt/ibm/ILOG/CPLEX_Studio2211/cplex")

然后再将 FindCPLEX.cmake 复制到/opt/scipoptsuite-9.1.0/scip/cmake/Modules 替换，再把 build 文件夹下内容删除，重新输入：

cmake .. -DIPOPT=false -DLPS=cpx -DTPI=tny

便不会报错，在日志内检查 CPLEX Found，即确定 Cplex 作为底层 LP 求解器

输入 sudo make，输入密码，进入冗长的编译过程，编译完不报错后输入 sudo make test 检查错误，由于没安装 IPOPT，所以会提示一些非线性模型错误，可以忽略。test 完成后输入 sudo make install 便会自动安装软件到/usr/local/bin，头文件和 lib 文件会安装到/usr/local/include 和/usr/local/lib 文件夹下，供 C/C++建模用。安装完成后输入 objdump -d /usr/local/bin/scip | grep ymm，然后就能看到标红的 ymm 寄存器，这是 AVX2 指令集对应的寄存器，证明面向 AVX2 指令集的优化编译已经完成。

```
1887019:       c4 62 ed b8 d9          vfmadd231pd %ymm1,%ymm2,%ymm11
188701e:       c4 e2 7d 19 51 90       vbroadcastsd -0x70(%rcx),%ymm2
1887024:       c4 e2 ed b8 f0          vfmadd231pd %ymm0,%ymm2,%ymm6
1887029:       c4 62 ed b8 e1          vfmadd231pd %ymm1,%ymm2,%ymm12
188702e:       c4 e2 7d 19 51 98       vbroadcastsd -0x68(%rcx),%ymm2
1887034:       c4 e2 ed b8 f8          vfmadd231pd %ymm0,%ymm2,%ymm7
1887039:       c4 62 ed b8 e9          vfmadd231pd %ymm1,%ymm2,%ymm13
188704c:       c5 fd 10 86 80 00 00    vmovupd 0x80(%rsi),%ymm0
1887054:       c5 fd 10 8e a0 00 00    vmovupd 0xa0(%rsi),%ymm1
188705c:       c4 e2 7d 19 51 80       vbroadcastsd -0x80(%rcx),%ymm2
1887062:       c4 e2 ed b8 e0          vfmadd231pd %ymm0,%ymm2,%ymm4
1887067:       c4 62 ed b8 d1          vfmadd231pd %ymm1,%ymm2,%ymm10
188706c:       c4 e2 7d 19 51 88       vbroadcastsd -0x78(%rcx),%ymm2
1887072:       c4 e2 ed b8 e8          vfmadd231pd %ymm0,%ymm2,%ymm5
1887077:       c4 62 ed b8 d9          vfmadd231pd %ymm1,%ymm2,%ymm11
188707c:       c4 e2 7d 19 51 90       vbroadcastsd -0x70(%rcx),%ymm2
```

7.SCIP 和 GCG 的使用

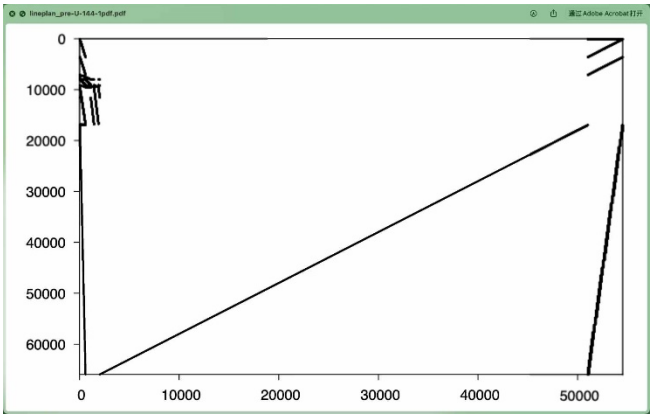
直接在命令行输入 scip 或者 gcg 即可进入求解器的交互界面，使用 read 命令读取模型，presolve 预处理模型，set 指令可以设置自己的策略，下面简单介绍 GCG(General Branch-and-Price & Column Generation Solver)，该求解器可以对线性连续和线性混合整数模型进行 Dantizg-Wolfe 和 Benders 自动化分解并求解，实测材料切割问题远远快于 Gurobi 和 Cplex。

该求解器默认为 Dantzig-Wolfe 分解，开启 Benders 探测和分解参阅：
https://gcg.or.rwth-aachen.de/doc-3.5.0/benders.html

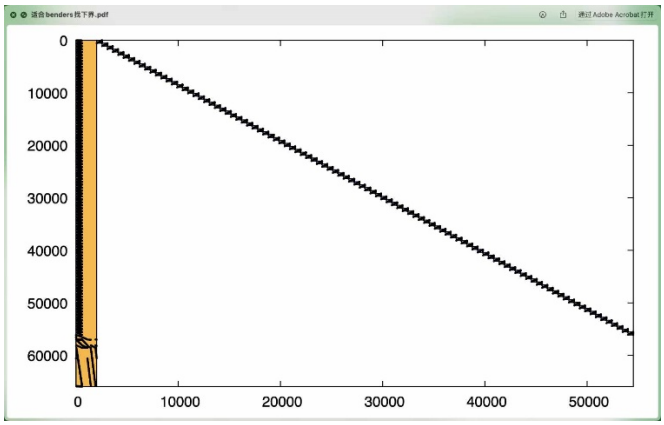
分解选择和图示化参阅：https://gcg.or.rwth-aachen.de/doc-3.5.0/explore-menu.html

以上网站有 SCIP 套件和 GCG 的详细文档，请自行探索。

分解效果，此为原问题：



分解后出现了典型的 benders 结构：



计算结果，下界提升明显：

```
GCG/explore> quit

GCG> set relaxing gcg mode 1
relaxing/gcg/mode = 1
GCG> read /Users/shengtianyi/Downloads/360s.sol

read problem </Users/shengtianyi/Downloads/360s.sol>
============

unknown variable <#> in line 1 of solution file </Users/shengtianyi/Downloads/360s.sol>
  (further unknown variables are ignored)
primal solution from solution file </Users/shengtianyi/Downloads/360s.sol> was accepted
original problem has 54526 variables (2026 bin, 52500 int, 0 impl, 0 cont) and 65952 constraints
GCG> opt


A Benders' decomposition is applied to solve the original problem.
Chosen structure has 9952 blocks, 2026 linking variables, and 0 linking constraints.
This decomposition has a maxwhite score of 0.954653.

  time | node  | left  |MLP iter|LP it/n| mem |mdpt |ovars|mvars|ocons|mcons|rows |mcuts|confs|strbr|  dualbound   | primalbound  |  gap
  61.1s|     1 |     0 |       0 |     - |3968M|   0 |  54k|   0 |  65k|   0 |  65k|   0 |   0 |   0 | 2.970450e+06 | 3.072504e+06 |  3.44%
```

非线性 MINLP 问题支持的运算符和求解效果：

```python
class Op:
    const = 'const'
    varidx = 'var'
    exp, log, sqrt, sin, cos = 'exp', 'log', 'sqrt', 'sin', 'cos'
    plus, minus, mul, div, power = '+', '-', '*', '/', '**'
    add = 'sum'
    prod = 'prod'
    fabs = 'abs'

Operator = Op()
```

```
SCIP/set> nlpi

  <ipopt>              Ipopt interface

SCIP/set/nlpi> ipopt

  hessian_approximation Indicates what Hessian information is to be used. Valid values if not empty: exact limited-memory []
  hsllib               Name of library containing HSL routines for load at runtime []
  linear_solver        Linear solver used for step computations. Valid values if not empty: ma27 ma57 ma77 ma86 ma97 pardiso mumps custom []
  linear_system_scaling Method for scaling the linear system. Valid values if not empty: none mc19 slack-based []
  mu_strategy          Update strategy for barrier parameter. Valid values if not empty: monotone adaptive []
  nlp_scaling_method   Select the technique used for scaling the NLP. Valid values if not empty: none user-scaling gradient-based equilibration-based []
  optfile              name of Ipopt options file []
  pardisolib           Name of library containing Pardiso routines (from pardiso-project.org) for load at runtime []
  print_level          Output verbosity level. -1 to use NLPI or Ipopt default. [-1]
  priority             priority of NLPI <ipopt> [1000]
  warm_start_push      amount (relative and absolute) by which starting point is moved away from bounds in warmstarts [1e-09]
```

```
presolving:
(round 1, fast)       103 del vars, 99 del conss, 0 add conss, 3466 chg bounds, 98 chg sides, 98 chg coeffs, 0 upgd conss, 0 impls, 198 clqs
(round 2, fast)       3569 del vars, 146 del conss, 0 add conss, 3466 chg bounds, 98 chg sides, 98 chg coeffs, 0 upgd conss, 0 impls, 198 clqs
(round 3, exhaustive) 3569 del vars, 1799 del conss, 0 add conss, 3466 chg bounds, 98 chg sides, 98 chg coeffs, 0 upgd conss, 0 impls, 198 clqs
(round 4, exhaustive) 3569 del vars, 1799 del conss, 0 add conss, 3466 chg bounds, 98 chg sides, 98 chg coeffs, 249 upgd conss, 0 impls, 198 clqs
   (5.8s) probing: 1000/6633 (15.1%) - 0 fixings, 0 aggregations, 2310 implications, 198 bound changes
   (5.8s) probing: 1001/6633 (15.1%) - 0 fixings, 0 aggregations, 2311 implications, 198 bound changes
   (5.8s) probing aborted: 1000/1000 successive useless probings
(round 5, exhaustive) 3569 del vars, 1799 del conss, 0 add conss, 3664 chg bounds, 98 chg sides, 0 chg coeffs, 249 upgd conss, 1279 impls, 1281 clqs
   (6.4s) probing: 1101/6633 (16.6%) - 0 fixings, 0 aggregations, 2428 implications, 198 bound changes
   (6.4s) probing aborted: 1000/1000 successive useless probings
   (6.4s) symmetry computation started: requiring (bin +, int +, cont +), (fixed: bin -, int -, cont -)
   (6.4s) no symmetry present (symcode time: 0.00)
presolving (6 rounds: 6 fast, 4 medium, 4 exhaustive):
 3569 deleted vars, 1799 deleted constraints, 0 added constraints, 3664 tightened bounds, 0 added holes, 98 changed sides, 0 changed coefficients
 1396 implications, 1281 cliques
presolved problem has 7031 variables (6633 bin, 0 int, 0 impl, 398 cont) and 8501 constraints
     51 constraints of type <varbound>
    198 constraints of type <setppc>
   7853 constraints of type <linear>
    399 constraints of type <nonlinear>
Presolving Time: 6.16

 time | node  | left  |LP iter|LP it/n|mem/heur|mdpt |vars |cons |rows |cuts |sepa|confs|strbr|  dualbound   | primalbound  |  gap   | compl.
  8.6s|     1 |     0 |   622 |     - |  1018M |   0 |7730 |8517 |8801 |   0 |  0 |  15 |   0 | 5.990476e+03 |      --      |    Inf | unknown
 10.1s|     1 |     0 |  3331 |     - |  1021M |   0 |7730 |8517 |9014 | 213 |  1 |  15 |   0 | 6.810871e+03 |      --      |    Inf | unknown
 10.4s|     1 |     0 |  3507 |     - |  1023M |   0 |7730 |8517 |9161 | 360 |  2 |  15 |   0 | 7.030084e+03 |      --      |    Inf | unknown
 10.7s|     1 |     0 |  3769 |     - |  1026M |   0 |7730 |8517 |9251 | 450 |  3 |  15 |   0 | 7.231198e+03 |      --      |    Inf | unknown
 11.0s|     1 |     0 |  3913 |     - |  1030M |   0 |7730 |8517 |9297 | 496 |  4 |  15 |   0 | 7.252167e+03 |      --      |    Inf | unknown
 11.3s|     1 |     0 |  4060 |     - |  1031M |   0 |7730 |8517 |9336 | 535 |  5 |  15 |   0 | 7.270341e+03 |      --      |    Inf | unknown
 11.6s|     1 |     0 |  4212 |     - |  1034M |   0 |7730 |8517 |9362 | 561 |  6 |  15 |   0 | 7.302934e+03 |      --      |    Inf | unknown
 11.8s|     1 |     0 |  4301 |     - |  1035M |   0 |7730 |8517 |9392 | 591 |  7 |  15 |   0 | 7.308147e+03 |      --      |    Inf | unknown
 12.1s|     1 |     0 |  4372 |     - |  1038M |   0 |7730 |8517 |9426 | 625 |  8 |  15 |   0 | 7.308147e+03 |      --      |    Inf | unknown
 12.4s|     1 |     0 |  4499 |     - |  1039M |   0 |7730 |8517 |9442 | 641 |  9 |  15 |   0 | 7.308897e+03 |      --      |    Inf | unknown
 13.2s|     1 |     0 |  4633 |     - |  1040M |   0 |7730 |8517 |9123 | 655 | 10 |  15 |   0 | 7.315197e+03 |      --      |    Inf | unknown
 13.6s|     1 |     0 |  4901 |     - |  1041M |   0 |7730 |8517 |9140 | 672 | 11 |  15 |   0 | 7.317862e+03 |      --      |    Inf | unknown
 13.8s|     1 |     0 |  5015 |     - |  1042M |   0 |7730 |8517 |9165 | 697 | 12 |  15 |   0 | 7.317862e+03 |      --      |    Inf | unknown
 14.1s|     1 |     0 |  5176 |     - |  1043M |   0 |7730 |8517 |9180 | 712 | 13 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 14.4s|     1 |     0 |  5224 |     - |  1044M |   0 |7730 |8517 |9193 | 725 | 14 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 time | node  | left  |LP iter|LP it/n|mem/heur|mdpt |vars |cons |rows |cuts |sepa|confs|strbr|  dualbound   | primalbound  |  gap   | compl.
 14.6s|     1 |     0 |  5262 |     - |  1045M |   0 |7730 |8517 |9202 | 734 | 15 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 15.6s|     1 |     0 |  5300 |     - |  1046M |   0 |7730 |8517 |8767 | 743 | 16 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 15.9s|     1 |     0 |  5316 |     - |  1046M |   0 |7730 |8517 |8771 | 747 | 17 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 16.1s|     1 |     0 |  5386 |     - |  1047M |   0 |7730 |8517 |8778 | 754 | 18 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 16.5s|     1 |     0 |  5413 |     - |  1048M |   0 |7730 |8517 |8790 | 766 | 19 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 16.7s|     1 |     0 |  5454 |     - |  1049M |   0 |7730 |8517 |8796 | 772 | 20 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 17.0s|     1 |     0 |  5583 |     - |  1049M |   0 |7730 |8517 |8804 | 780 | 21 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 18.4s|     1 |     0 |  5659 |     - |  1050M |   0 |7730 |8517 |8696 | 786 | 22 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
 18.7s|     1 |     0 |  5712 |     - |  1050M |   0 |7730 |8517 |8704 | 794 | 23 |  15 |   0 | 7.318926e+03 |      --      |    Inf | unknown
```

同一个 MINLP 问题，并行化求解：

time	mem	dualbound	primalbound	gap
$ 300s	1980M	7.421893e+03	9.263486e+04	1148.13%
$ 483s	2325M	7.426828e+03	1.228218e+04	65.38%
$ 559s	2937M	7.426828e+03	8.574798e+03	15.46%
$ 628s	3315M	7.433856e+03	8.528615e+03	14.73%
$ 716s	3680M	7.444320e+03	7.993501e+03	7.38%
$ 808s	3818M	7.445647e+03	7.607829e+03	2.18%
$ 855s	3950M	7.450898e+03	7.584381e+03	1.79%
888s	4020M	7.473841e+03	7.584381e+03	1.48%